# Robotics Lab: Homework 1

Building your robot manipulator

**FALCO PIETRO** **Matricola: P38000208**

**GRUPPO: Davide Busco, Pietro Falco, Davide Rubinacci, Giuseppe Saggese**

This document contains the homwework 1 of the Robotics Lab class.

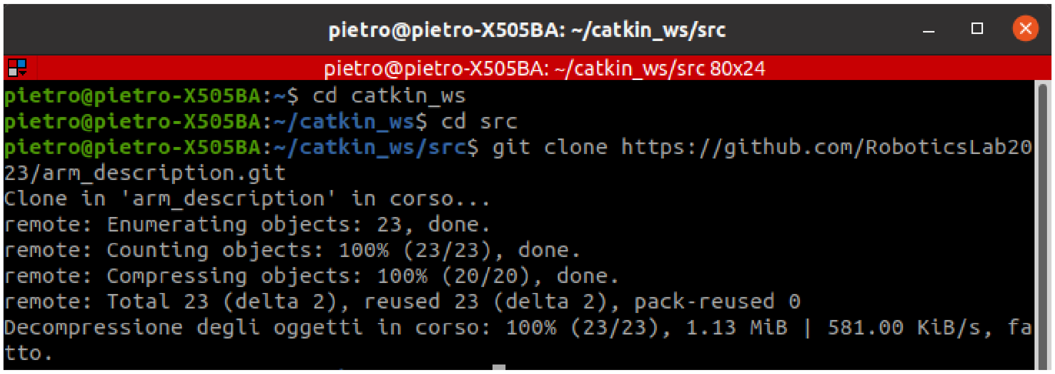
## Building your robot manipulator

The goal of this homework is to build ROS packages to simulate a 4-degrees-of-freedom robotic manipulator arm into the Gazebo environment. The student is requested to address the following points and provide a detailed report of the employed methods. In addition, a personal github repo with all the developed code must be shared with the instuctor. The report is due in one week from the homewerk release.

1 Create the description of your robot and visualize it in Rviz

1.a Download the *arm_description* package from the repo *https://github.com/RoboticsLab2023/arm_description.git* into your *catkin_ws* using *git* commands

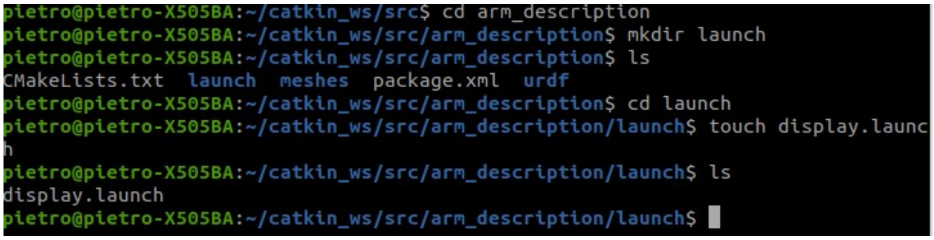
**terminal:**


pietro@pietro-X505BA: ~/catkin_ws/src
pietro@pietro-X505BA: ~/catkin_ws/src 80x24

```
pietro@pietro-X505BA:~$ cd catkin_ws
pietro@pietro-X505BA:~/catkin_ws$ cd src
pietro@pietro-X505BA:~/catkin_ws/src$ git clone https://github.com/RoboticsLab20
23/arm_description.git
Clone in 'arm_description' in corso...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 23 (delta 2), reused 23 (delta 2), pack-reused 0
Decompressione degli oggetti in corso: 100% (23/23), 1.13 MiB | 581.00 KiB/s, fa
tto.
```

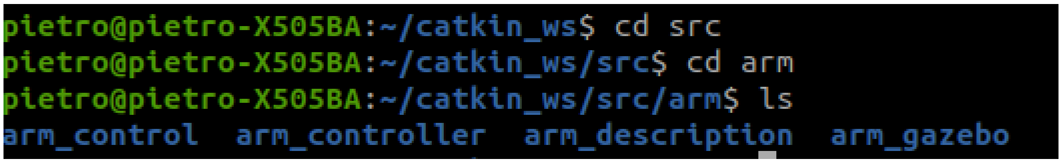
1.b Within the package create a *launch* folder containing a launch file named *display.launch* that loads the URDF as a *robot_description* ROS param and starts the *robot_state_publisher* node, the *joint_state_publisher* node, and the *rviz* node. Launch the file using *roslaunch*. **Note:** To visualize your robot in rviz you have to changhe the Fixed Frame in the lateral bar and add the *RobotModel* plugin interface. **Optional:** save a *.rviz* configuration file, thad automatically loads the *RobotModel* plugin by default, and give it as an argument to your node in the *display.launch* file

**terminal:** I created a launch folder with the command "mkdir" then I create a file-launch with command "touch" in the arm_description package



```
pietro@pietro-X505BA:~/catkin_ws/src$ cd arm_description
pietro@pietro-X505BA:~/catkin_ws/src/arm_description$ mkdir launch
pietro@pietro-X505BA:~/catkin_ws/src/arm_description$ ls
CMakeLists.txt  launch  meshes  package.xml  urdf
pietro@pietro-X505BA:~/catkin_ws/src/arm_description$ cd launch
pietro@pietro-X505BA:~/catkin_ws/src/arm_description/launch$ touch display.launc
h
pietro@pietro-X505BA:~/catkin_ws/src/arm_description/launch$ ls
display.launch
pietro@pietro-X505BA:~/catkin_ws/src/arm_description/launch$ 
```

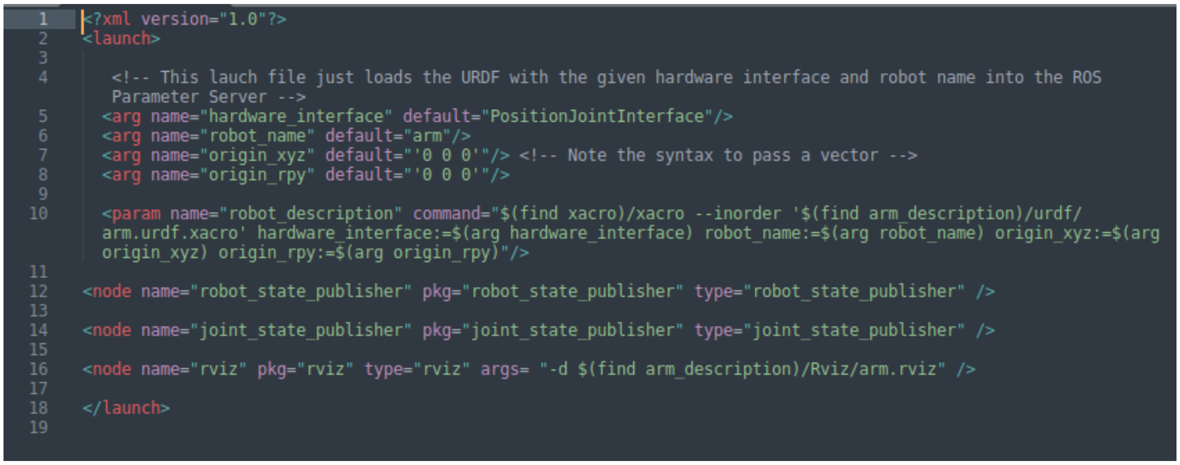
In the end I move everything in a single "arm" folder for a matter of compactness:



```
pietro@pietro-X505BA:~/catkin_ws$ cd src
pietro@pietro-X505BA:~/catkin_ws/src$ cd arm
pietro@pietro-X505BA:~/catkin_ws/src/arm$ ls
arm_control  arm_controller  arm_description  arm_gazebo
```

**display.launch:**

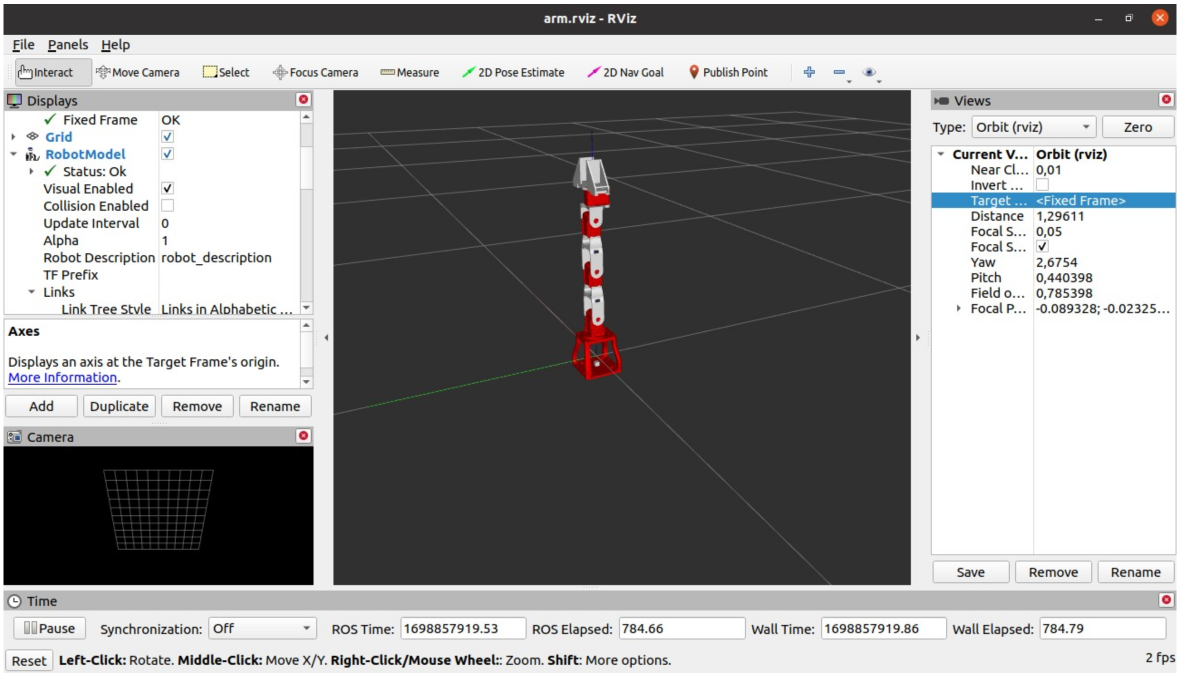
this file loads the urdf.xacro.file as a robot_description ROS param and starts the robot_state_publisher node, the joint_state_publisher node and the rviz node. robot_state_publisher is a node from the robot_state_publisher package that publishes the transformation between robot links based on the robot description in the "robot_description" parameter. Joint_state_publisher is a node from the joint_state_publisher package that allows publishing the robot's joint state, so that other parts of the system can know the joint positions. rviz is a node from the rviz package that starts the 3D visualization tool RViz with a configuration specified by a .rviz file.



```
<?xml version="1.0"?>
<launch>

   <!-- This lauch file just loads the URDF with the given hardware interface and robot name into the ROS
   Parameter Server -->
  <arg name="hardware_interface" default="PositionJointInterface"/>
  <arg name="robot_name" default="arm"/>
  <arg name="origin_xyz" default="'0 0 0'"/> <!-- Note the syntax to pass a vector -->
  <arg name="origin_rpy" default="'0 0 0'"/>

  <param name="robot_description" command="$(find xacro)/xacro --inorder '$(find arm_description)/urdf/
  arm.urdf.xacro' hardware_interface:=$(arg hardware_interface) robot_name:=$(arg robot_name) origin_xyz:=$(arg
  origin_xyz) origin_rpy:=$(arg origin_rpy)"/>

<node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />

<node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />

<node name="rviz" pkg="rviz" type="rviz" args= "-d $(find arm_description)/Rviz/arm.rviz" />

</launch>

```

**rviz:**

“roslaunch arm_description display.launch” by terminal


arm.rviz - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Add Duplicate Remove Rename
Camera
Views
Save Remove Rename
Time
Pause
Reset

Fixed Frame OK
Grid ☑
RobotModel ☑
Status: Ok
Visual Enabled ☑
Collision Enabled □
Update Interval 0
Alpha 1
Robot Description robot_description
TF Prefix
Links
Link Tree Style Links in Alphabetic ...

Axes

Displays an axis at the Target Frame's origin. More Information.

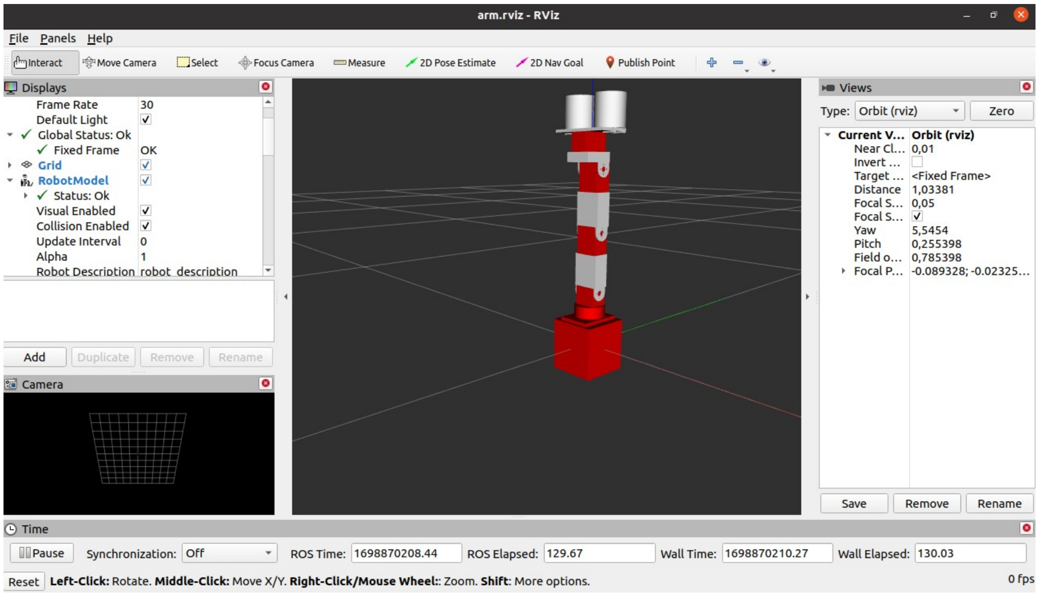
Type: Orbit (rviz) Zero
Current V... Orbit (rviz)
Near Cl... 0,01
Invert ... □
Target ... <Fixed Frame>
Distance 1,29611
Focal S... 0,05
Focal S... ☑
Yaw 2,6754
Pitch 0,440398
Field o... 0,785398
Focal P... -0.089328; -0.02325...

Synchronization: Off ROS Time: 1698857919.53 ROS Elapsed: 784.66 Wall Time: 1698857919.86 Wall Elapsed: 784.79

**Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 2 fps

1.c Substitute the collision meshes of your URDF with primitive shapes. Use *<box>* geometries of reasonabe size approximating the links. **Hint:** Enable collision visualization in rviz (go to the lateral bar > Robot model > Collision Enabled) to adjust the collision meshes size

I substitute the collision meshes of every links with the box and cylinder tags and I chose the correct dimensions of the their shapes and after I launch display.launch:


arm.rviz - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Add Duplicate Remove Rename
Camera
Views
Save Remove Rename
Time
Pause
Reset

Frame Rate 30
Default Light ☑
Global Status: Ok
Fixed Frame OK
Grid ☑
RobotModel ☑
Status: Ok
Visual Enabled ☑
Collision Enabled ☑
Update Interval 0
Alpha 1
Robot Description robot_description

Type: Orbit (rviz) Zero
Current V... Orbit (rviz)
Near Cl... 0,01
Invert ... □
Target ... <Fixed Frame>
Distance 1,03381
Focal S... 0,05
Focal S... ☑
Yaw 5,5454
Pitch 0,255398
Field o... 0,785398
Focal P... -0.089328; -0.02325...

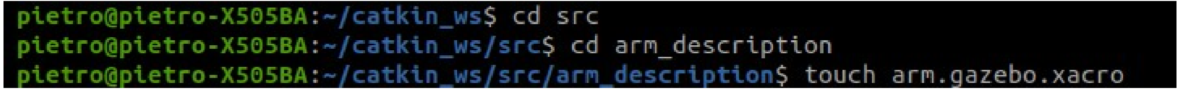
Synchronization: Off ROS Time: 1698870208.44 ROS Elapsed: 129.67 Wall Time: 1698870210.27 Wall Elapsed: 130.03

**Left-Click:** Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 0 fps

1.d Create a file named *arm.gazebo.xacro* within your package, define a *xacro:macro* inside your file containing all the *<gazebo>* tags you find within your *arm.urdf* and import it in your URDF using *xacro:include*. Remember to rename your URDF file to *arm.urdf.xacro*, add the string *xmlns:xacro="http://www.ros.org/wiki/xacro"* within the *<robot>* tag, and load the URDF in your launch file using the *xacro* routine

**terminal:**

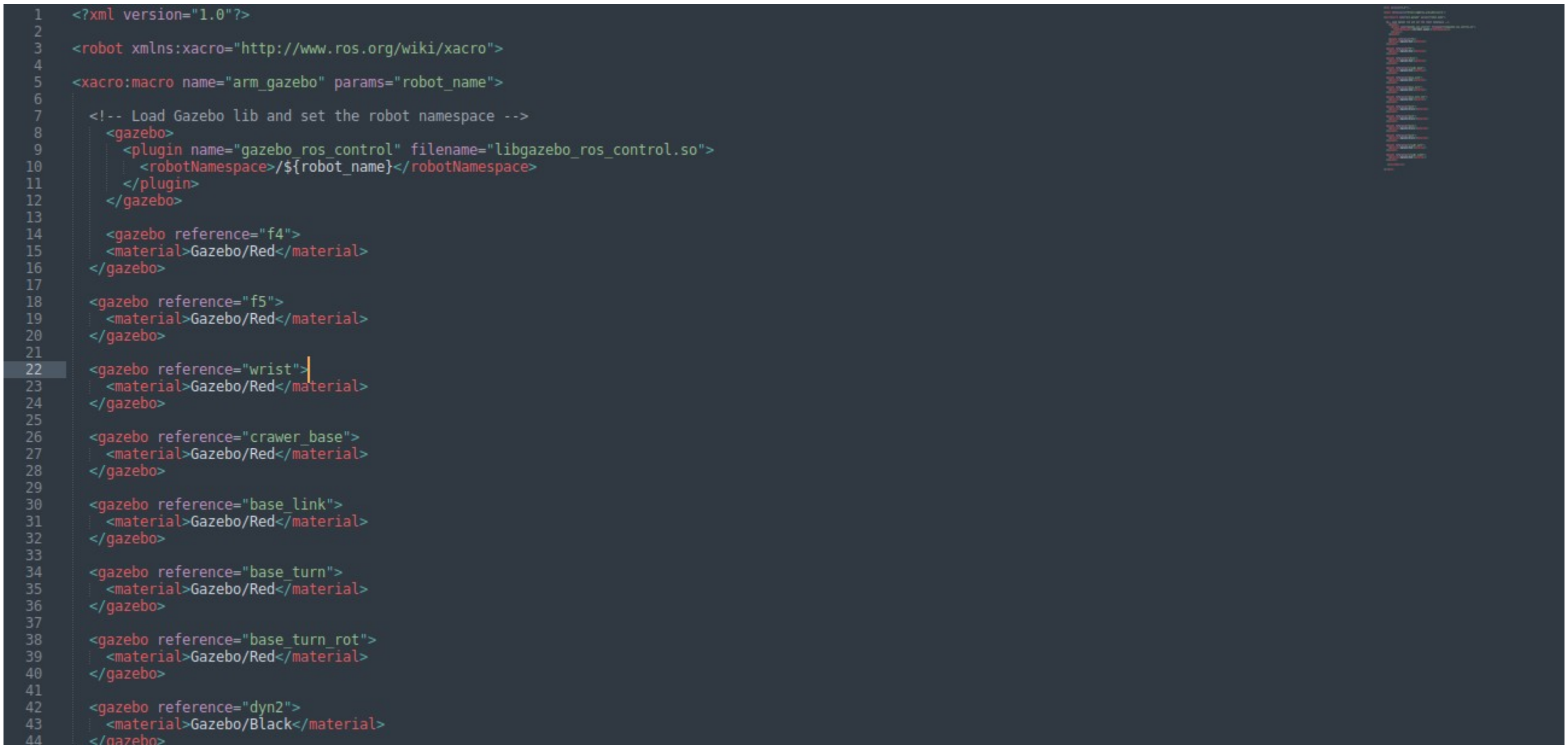
I create il file arm.gazebo.xacro using the command "touch"



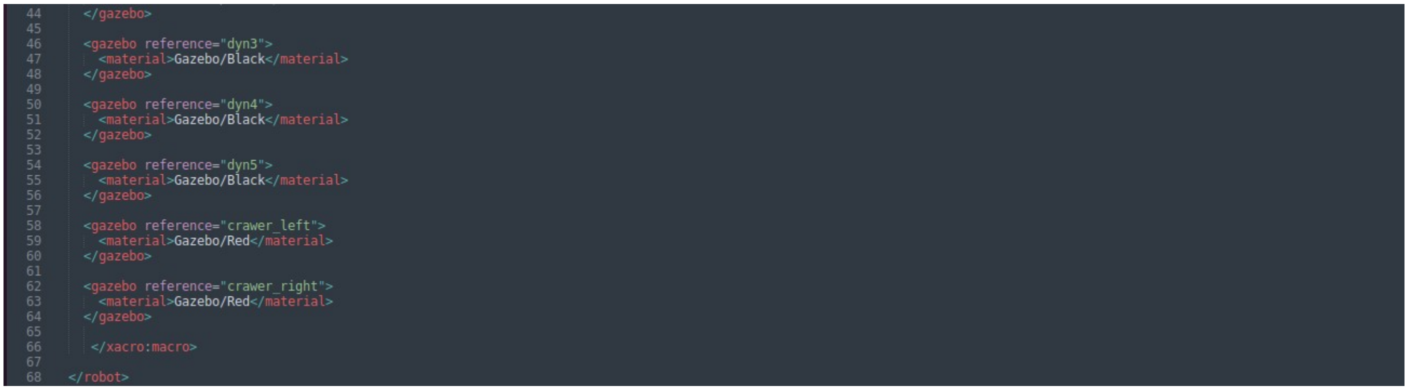
```
pietro@pietro-X505BA:~/catkin_ws$ cd src
pietro@pietro-X505BA:~/catkin_ws/src$ cd arm_description
pietro@pietro-X505BA:~/catkin_ws/src/arm_description$ touch arm.gazebo.xacro
```

**arm.gazebo.xacro:**

I added  all lines regarding <gazebo> tags that they were in arm.urdf.xacro in arm.gazebo.xacro



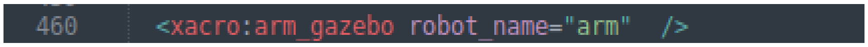
```
<?xml version="1.0"?>

<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

<xacro:macro name="arm_gazebo" params="robot_name">

  <!-- Load Gazebo lib and set the robot namespace -->
    <gazebo>
      <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
        <robotNamespace>/${robot_name}</robotNamespace>
      </plugin>
    </gazebo>

    <gazebo reference="f4">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="f5">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="wrist">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="crawer_base">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="base_link">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="base_turn">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="base_turn_rot">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="dyn2">
    <material>Gazebo/Black</material>
```



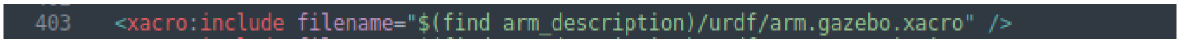
```
  </gazebo>

  <gazebo reference="dyn3">
    <material>Gazebo/Black</material>
  </gazebo>

  <gazebo reference="dyn4">
    <material>Gazebo/Black</material>
  </gazebo>

  <gazebo reference="dyn5">
    <material>Gazebo/Black</material>
  </gazebo>

  <gazebo reference="crawer_left">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="crawer_right">
    <material>Gazebo/Red</material>
  </gazebo>

   </xacro:macro>

</robot>
```

**arm.urdf.xacro:**

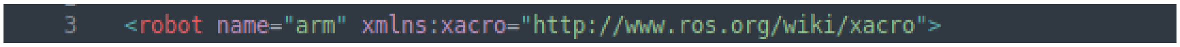
I substituted all lines regarding <gazebo> tags in arm.urdf.xacro with this lines



```
<xacro:arm_gazebo robot_name="arm"  />
```



```
<xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />
```

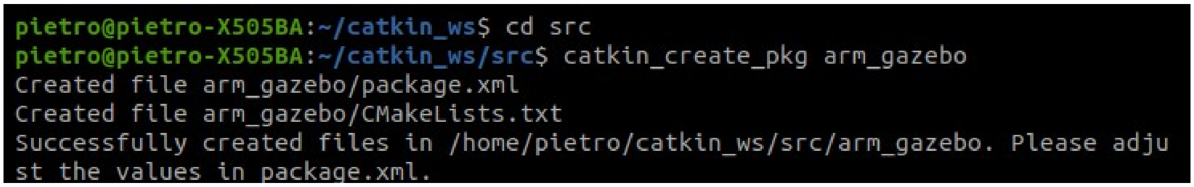


```
<robot name="arm" xmlns:xacro="http://www.ros.org/wiki/xacro">
```

2 Add transmission and controllers to your robot and spawn it in Gazebo

2.a Create a package named *arm_gazebo*

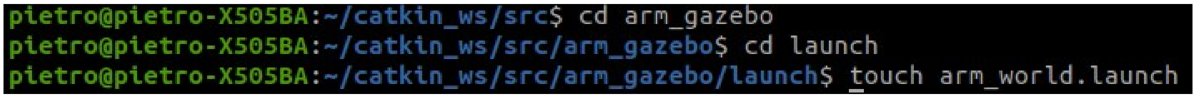
**terminal:** *I create the package arm_gazebo using command "catkin_create_pkg package_name"*



```
pietro@pietro-X505BA:~/catkin_ws$ cd src
pietro@pietro-X505BA:~/catkin_ws/src$ catkin_create_pkg arm_gazebo
Created file arm_gazebo/package.xml
Created file arm_gazebo/CMakeLists.txt
Successfully created files in /home/pietro/catkin_ws/src/arm_gazebo. Please adju
st the values in package.xml.
```

2.b Within this package create a *launch* folder containing a *arm_world.launch* file

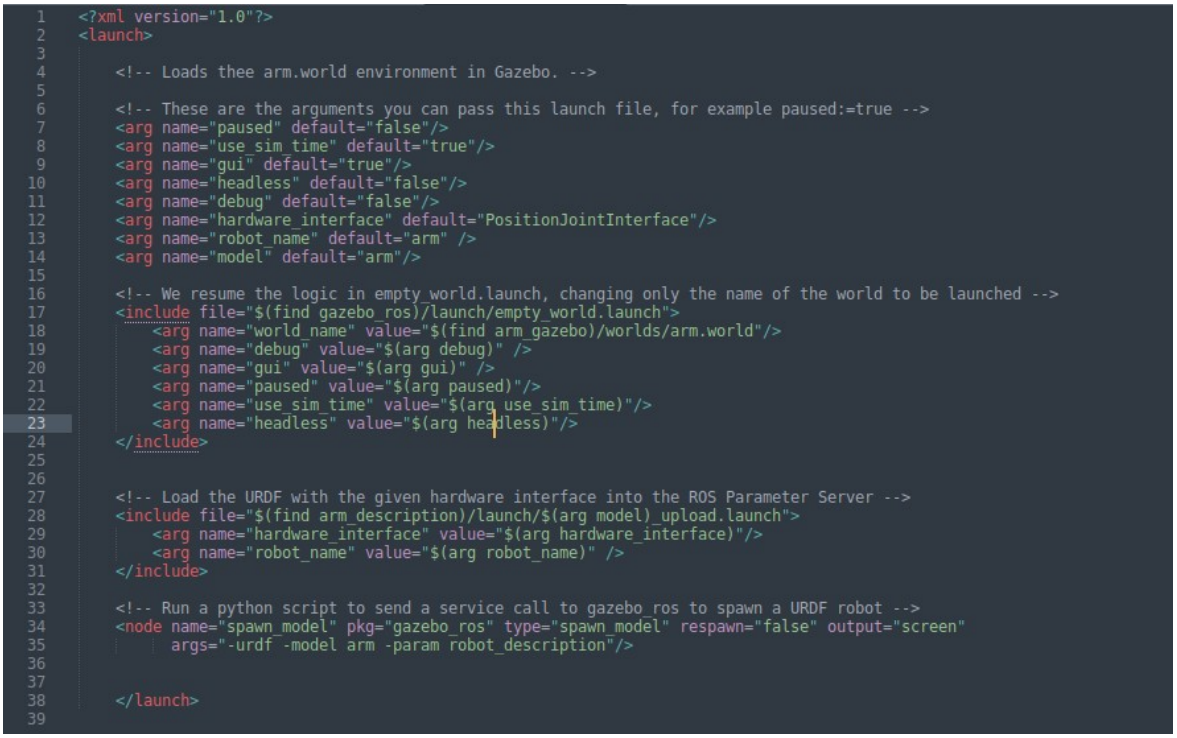
**terminal:** I create the folder launch with the command "mkdir" and the file launch with the command "touch"



```
pietro@pietro-X505BA:~/catkin_ws/src$ cd arm_gazebo
pietro@pietro-X505BA:~/catkin_ws/src/arm_gazebo$ cd launch
pietro@pietro-X505BA:~/catkin_ws/src/arm_gazebo/launch$ touch arm_world.launch
```

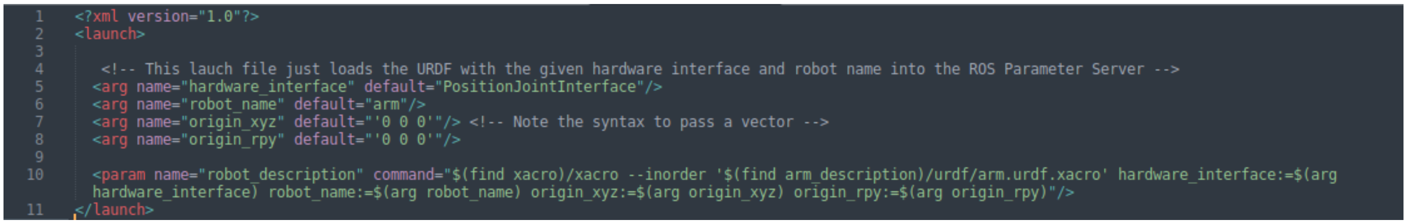
2.c Fill this launch file with commands that load the URDF into the ROS Parameter Server and spawn your robot using the *spawn_model* node. **Hint:** follow the *iiwa_world.launch* example from the package *iiwa_stack*: *https://github.com/IFL-CAMP/iiwa_stack/tree/master*. Launch the *arm_world.launch* file to visualize the robot in Gazebo

**arm_world.launch:** This launch file is used to set up a Gazebo simulation environment for a robot in ROS. It loads a specific world, sets up the robot's description, and spawns the robot model into Gazebo. The launch file allows for various configuration options, such as pausing the simulation, using simulated time, and controlling the GUI display, among other things.



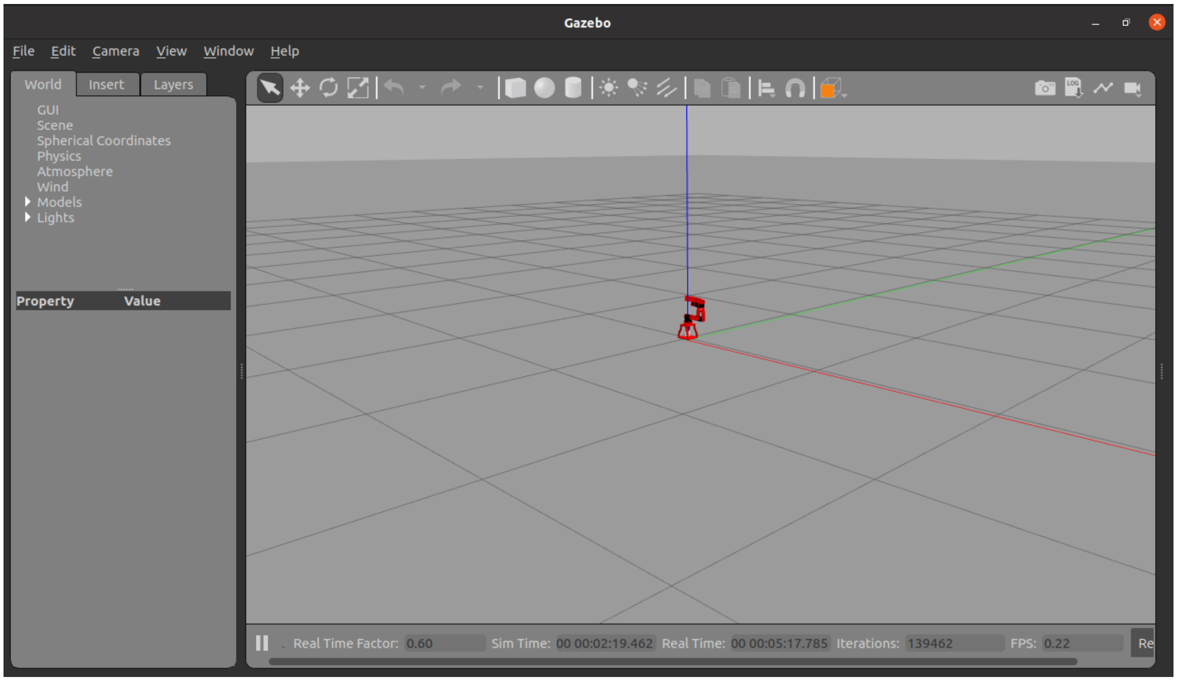
```
<?xml version="1.0"?>
<launch>

    <!-- Loads thee arm.world environment in Gazebo. -->

    <!-- These are the arguments you can pass this launch file, for example paused:=true -->
    <arg name="paused" default="false"/>
    <arg name="use_sim_time" default="true"/>
    <arg name="gui" default="true"/>
    <arg name="headless" default="false"/>
    <arg name="debug" default="false"/>
    <arg name="hardware_interface" default="PositionJointInterface"/>
    <arg name="robot_name" default="arm" />
    <arg name="model" default="arm"/>

    <!-- We resume the logic in empty_world.launch, changing only the name of the world to be launched -->
    <include file="$(find gazebo_ros)/launch/empty_world.launch">
        <arg name="world_name" value="$(find arm_gazebo)/worlds/arm.world"/>
        <arg name="debug" value="$(arg debug)" />
        <arg name="gui" value="$(arg gui)" />
        <arg name="paused" value="$(arg paused)"/>
        <arg name="use_sim_time" value="$(arg use_sim_time)"/>
        <arg name="headless" value="$(arg headless)"/>
    </include>


    <!-- Load the URDF with the given hardware interface into the ROS Parameter Server -->
    <include file="$(find arm_description)/launch/$(arg model)_upload.launch">
        <arg name="hardware_interface" value="$(arg hardware_interface)"/>
        <arg name="robot_name" value="$(arg robot_name)" />
    </include>

    <!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
    <node name="spawn_model" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
          args="-urdf -model arm -param robot_description"/>


    </launch>

```

**arm_upload.launch:**



```
<?xml version="1.0"?>
<launch>

   <!-- This lauch file just loads the URDF with the given hardware interface and robot name into the ROS Parameter Server -->
  <arg name="hardware_interface" default="PositionJointInterface"/>
  <arg name="robot_name" default="arm"/>
  <arg name="origin_xyz" default="'0 0 0'"/> <!-- Note the syntax to pass a vector -->
  <arg name="origin_rpy" default="'0 0 0'"/>

  <param name="robot_description" command="$(find xacro)/xacro --inorder '$(find arm_description)/urdf/arm.urdf.xacro' hardware_interface:=$(arg
  hardware_interface) robot_name:=$(arg robot_name) origin_xyz:=$(arg origin_xyz) origin_rpy:=$(arg origin_rpy)"/>
</launch>
```

I used command “roslaunch arm_gazebo arm_world.launch” **by terminal** to launch arm_world.launch

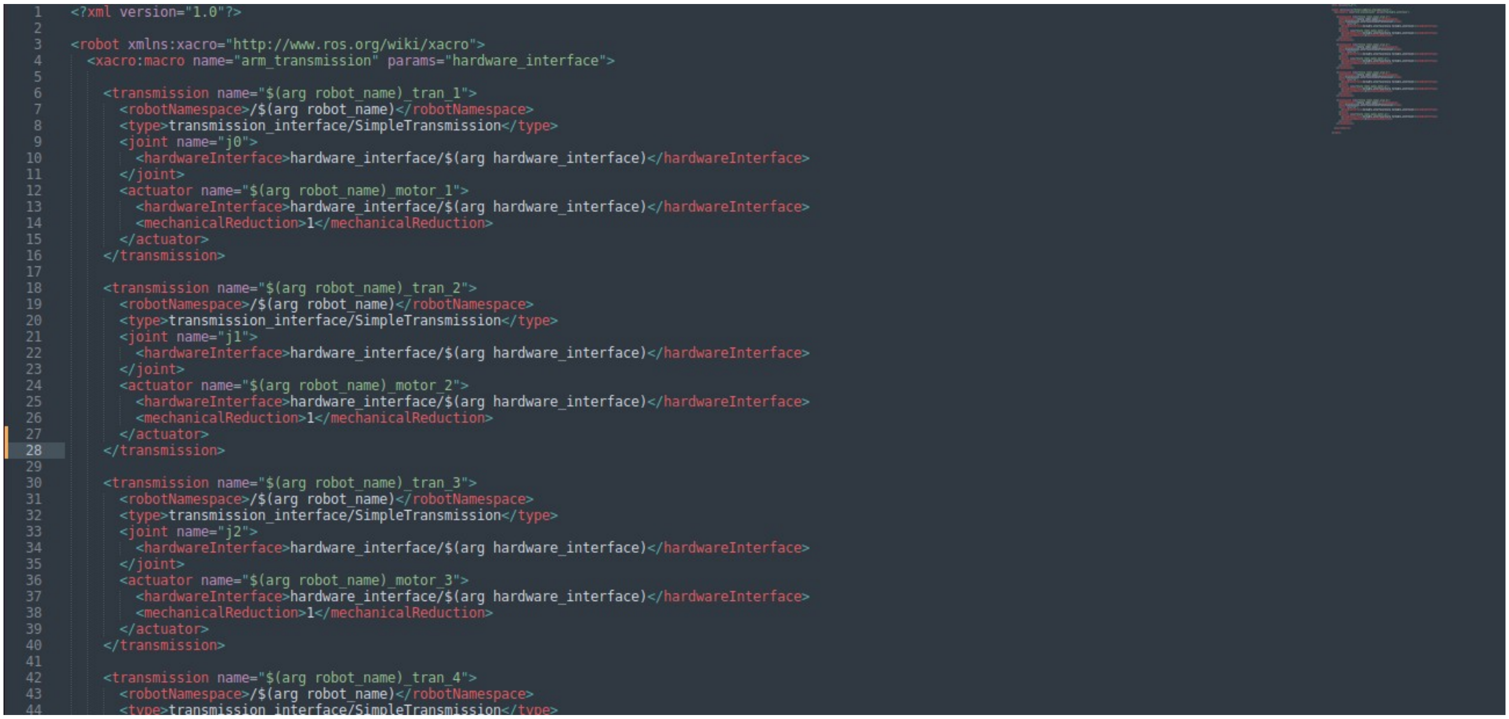


Real Time Factor: 0.60 Sim Time: 00 00:02:19.462 Real Time: 00 00:05:17.785 Iterations: 139462 FPS: 0.22

How we can see after few minutes the robot collapses on itself

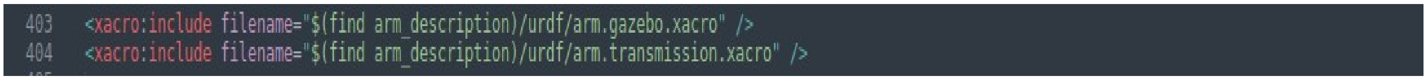
2.d Now add a *PositionJointInterface* as hardware interface to your robot: create a *arm.transmission.xacro* file into your *arm_description/urdf* folder containing a *xacro:macro* with the hardware interface and load it into your *arm.urdf.xacro* file using *xacro:include*. Launch the file

**arm.transmission.xacro:** is a URDF file written using the XACRO language, which is commonly used for generating robot descriptions in ROS. The file defines four transmissions for the robot and uses a macro called "arm_transmission" to simplify the definition of these transmissions.

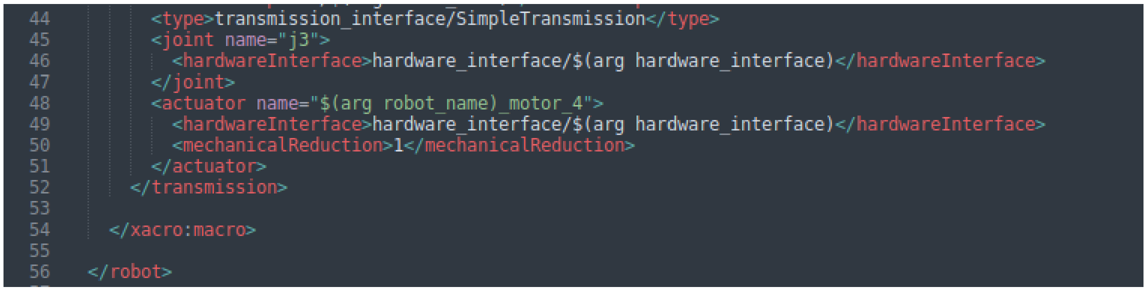


```
<?xml version="1.0"?>

<robot xmlns:xacro="http://www.ros.org/wiki/xacro">
  <xacro:macro name="arm_transmission" params="hardware_interface">

    <transmission name="$(arg robot_name)_tran_1">
      <robotNamespace>/$(arg robot_name)</robotNamespace>
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="j0">
        <hardwareInterface>hardware_interface/$(arg hardware_interface)</hardwareInterface>
      </joint>
      <actuator name="$(arg robot_name)_motor_1">
        <hardwareInterface>hardware_interface/$(arg hardware_interface)</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

    <transmission name="$(arg robot_name)_tran_2">
      <robotNamespace>/$(arg robot_name)</robotNamespace>
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="j1">
        <hardwareInterface>hardware_interface/$(arg hardware_interface)</hardwareInterface>
      </joint>
      <actuator name="$(arg robot_name)_motor_2">
        <hardwareInterface>hardware_interface/$(arg hardware_interface)</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

    <transmission name="$(arg robot_name)_tran_3">
      <robotNamespace>/$(arg robot_name)</robotNamespace>
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="j2">
        <hardwareInterface>hardware_interface/$(arg hardware_interface)</hardwareInterface>
      </joint>
      <actuator name="$(arg robot_name)_motor_3">
        <hardwareInterface>hardware_interface/$(arg hardware_interface)</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

    <transmission name="$(arg robot_name)_tran_4">
      <robotNamespace>/$(arg robot_name)</robotNamespace>
      <type>transmission_interface/SimpleTransmission</type>
```

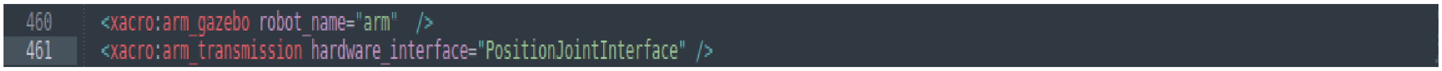
**arm.urdf.xacro:** I added *xacro:macro* with the hardware interface and load it into your *arm.urdf.xacro* file using *xacro:include in arm.urdf.xacro*.



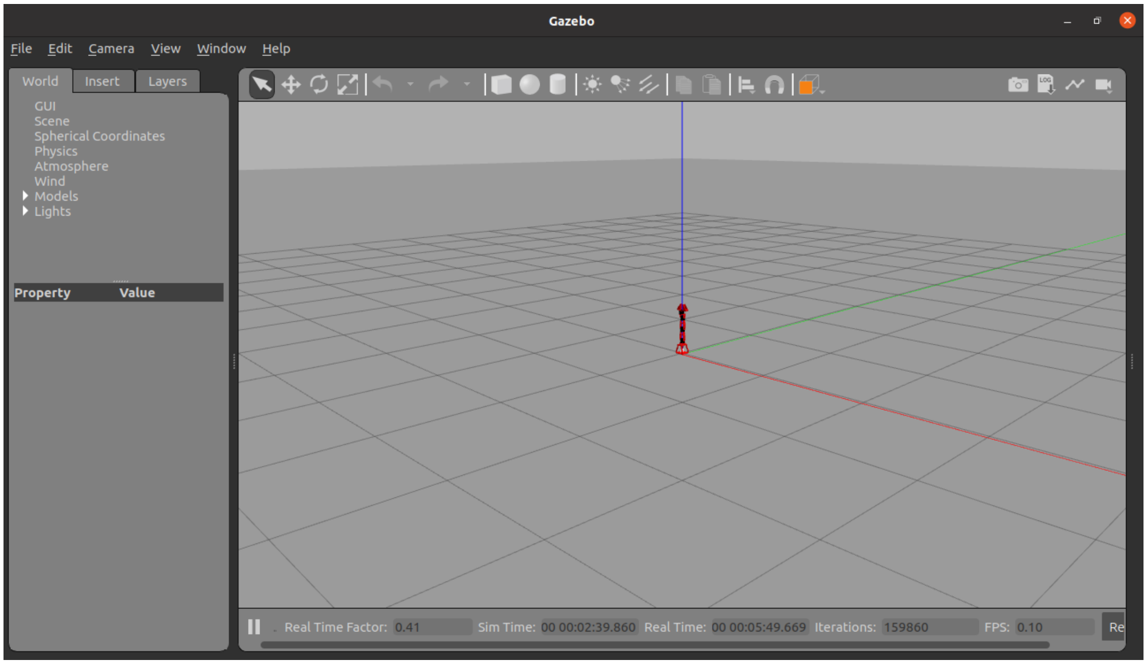
```
<xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />
<xacro:include filename="$(find arm_description)/urdf/arm.transmission.xacro" />
```



```
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="j3">
        <hardwareInterface>hardware_interface/$(arg hardware_interface)</hardwareInterface>
      </joint>
      <actuator name="$(arg robot_name)_motor_4">
        <hardwareInterface>hardware_interface/$(arg hardware_interface)</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

  </xacro:macro>

</robot>
```



```
<xacro:arm_gazebo robot_name="arm"  />
<xacro:arm_transmission hardware_interface="PositionJointInterface" />
```

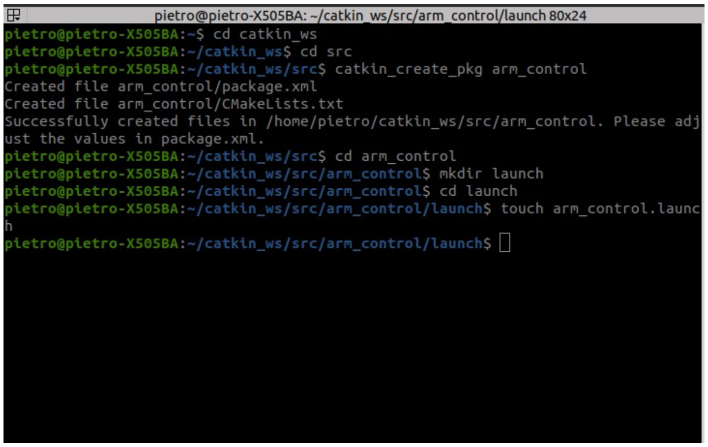


Real Time Factor: 0.41 Sim Time: 00 00:02:39.860 Real Time: 00 00:05:49.669 Iterations: 159860 FPS: 0.10

How we can see from simulation time, the robot is maintained in the same position without falling for the force of gravity compared to the previous case where i have not implemented arm_trasmission

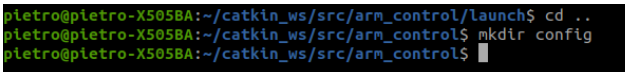
2.e Add joint position controllers to your robot: create a *arm_control* package with a *arm_control.launch* file inside its *launch* folder and a *arm_control.yaml* file within its *config* folder

**terminal:**

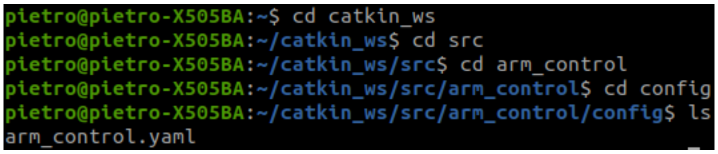


pietro@pietro-X505BA: ~/catkin_ws/src/arm_control/launch 80x24

```
pietro@pietro-X505BA:~$ cd catkin_ws
pietro@pietro-X505BA:~/catkin_ws$ cd src
pietro@pietro-X505BA:~/catkin_ws/src$ catkin_create_pkg arm_control
Created file arm_control/package.xml
Created file arm_control/CMakeLists.txt
Successfully created files in /home/pietro/catkin_ws/src/arm_control. Please adj
ust the values in package.xml.
pietro@pietro-X505BA:~/catkin_ws/src$ cd arm_control
pietro@pietro-X505BA:~/catkin_ws/src/arm_control$ mkdir launch
pietro@pietro-X505BA:~/catkin_ws/src/arm_control$ cd launch
pietro@pietro-X505BA:~/catkin_ws/src/arm_control/launch$ touch arm_control.launc
h
pietro@pietro-X505BA:~/catkin_ws/src/arm_control/launch$ 
```



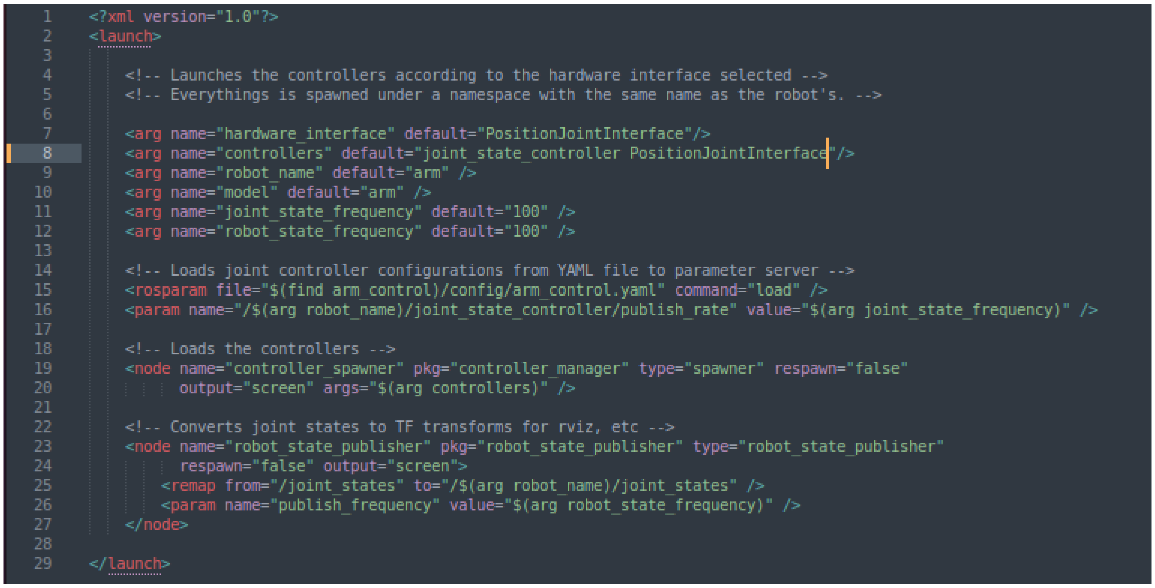
```
pietro@pietro-X505BA:~/catkin_ws/src/arm_control/launch$ cd ..
pietro@pietro-X505BA:~/catkin_ws/src/arm_control$ mkdir config
pietro@pietro-X505BA:~/catkin_ws/src/arm_control$ 
```



```
pietro@pietro-X505BA:~$ cd catkin_ws
pietro@pietro-X505BA:~/catkin_ws$ cd src
pietro@pietro-X505BA:~/catkin_ws/src$ cd arm_control
pietro@pietro-X505BA:~/catkin_ws/src/arm_control$ cd config
pietro@pietro-X505BA:~/catkin_ws/src/arm_control/config$ ls
arm_control.yaml
```

2.f  Fill the *arm_control.launch* file with commands that load the joint controller configurations from the *.yaml* file to the parameter server and spawn the controllers using the *controller_manager* package. **Hint:** follow the *iiwa_control.launch* example from corresponding package

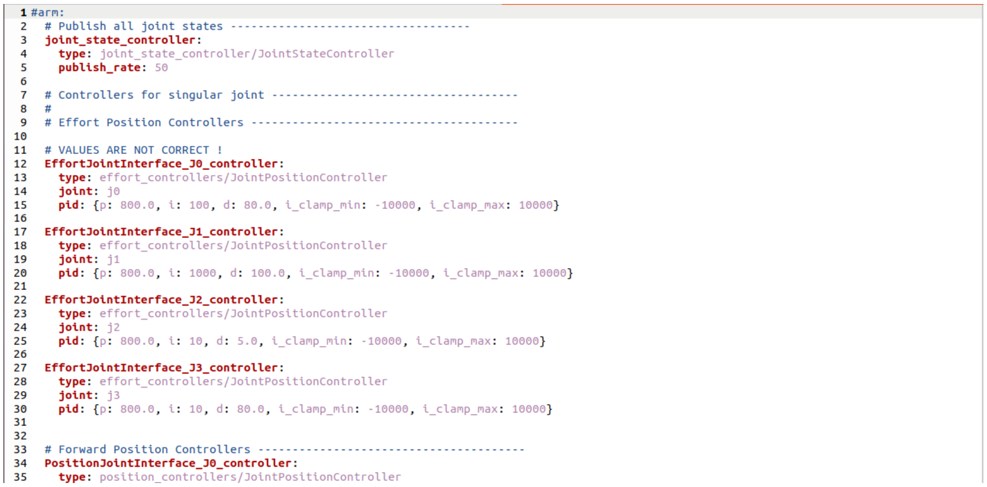
**arm_control.launch:** this launch file configures and starts controllers for robot control, loads controller configurations from a YAML file, and starts the "robot_state_publisher" node to publish the robot's joint states**.** To make the gazebo communicates with joint_states we must to change the remap in < remap from "/joint_states" to "/$(arg robot_name)/joint_states" /> .



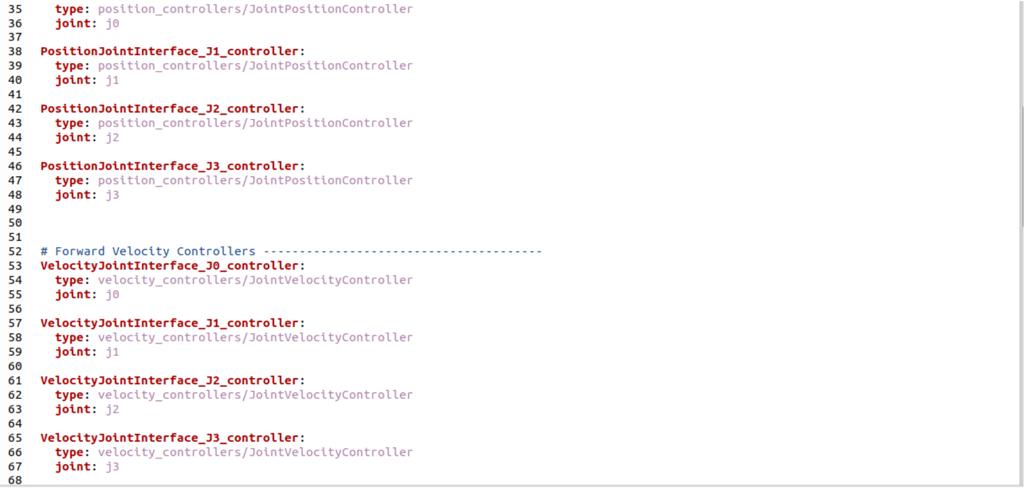
```xml
<?xml version="1.0"?>
<launch>

    <!-- Launches the controllers according to the hardware interface selected -->
    <!-- Everythings is spawned under a namespace with the same name as the robot's. -->

    <arg name="hardware_interface" default="PositionJointInterface"/>
    <arg name="controllers" default="joint_state_controller PositionJointInterface"/>
    <arg name="robot_name" default="arm" />
    <arg name="model" default="arm" />
    <arg name="joint_state_frequency" default="100" />
    <arg name="robot_state_frequency" default="100" />

    <!-- Loads joint controller configurations from YAML file to parameter server -->
    <rosparam file="$(find arm_control)/config/arm_control.yaml" command="load" />
    <param name="/$(arg robot_name)/joint_state_controller/publish_rate" value="$(arg joint_state_frequency)" />

    <!-- Loads the controllers -->
    <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
          output="screen" args="$(arg controllers)" />

    <!-- Converts joint states to TF transforms for rviz, etc -->
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
          respawn="false" output="screen">
        <remap from="/joint_states" to="/$(arg robot_name)/joint_states" />
        <param name="publish_frequency" value="$(arg robot_state_frequency)" />
    </node>

</launch>
```

(g) Fill the arm *arm_control.yaml* adding a *joint_state_controller* and a *JointPositionController* to all the joint

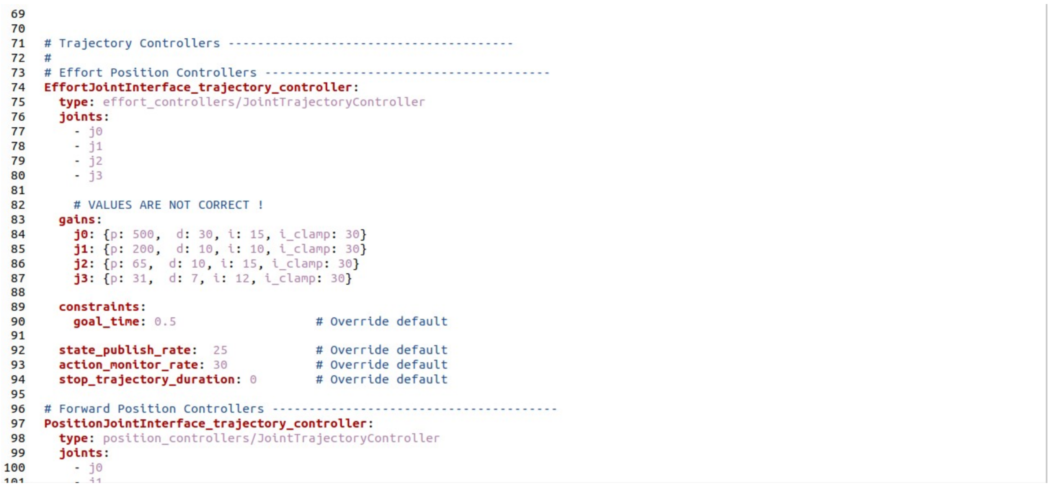
**arm_control.yaml:**



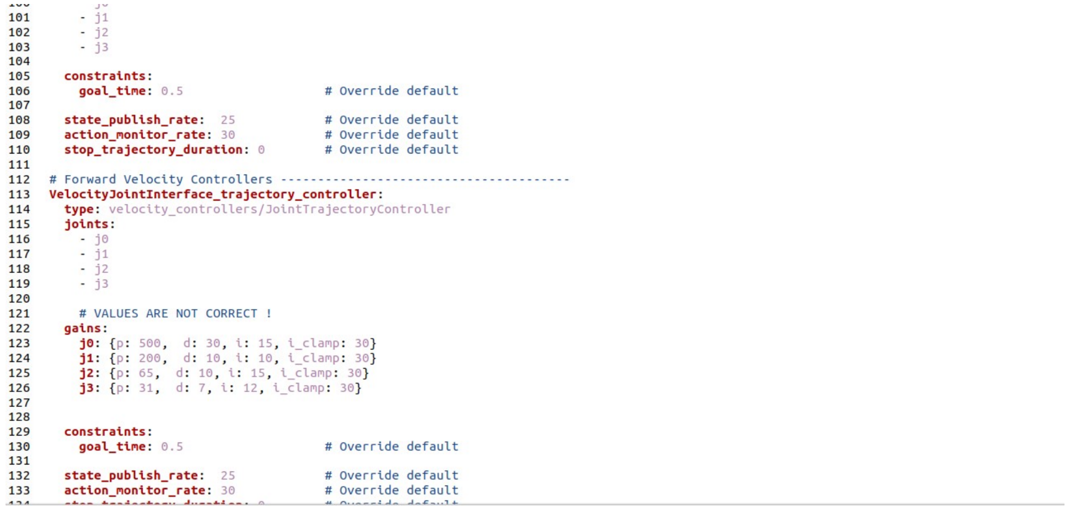
```yaml
#arm:
  # Publish all joint states -----------------------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50

  # Controllers for singular joint ------------------------------------
  #
  # Effort Position Controllers ---------------------------------------

  # VALUES ARE NOT CORRECT !
  EffortJointInterface_J0_controller:
    type: effort_controllers/JointPositionController
    joint: j0
    pid: {p: 800.0, i: 100, d: 80.0, i_clamp_min: -10000, i_clamp_max: 10000}

  EffortJointInterface_J1_controller:
    type: effort_controllers/JointPositionController
    joint: j1
    pid: {p: 800.0, i: 1000, d: 100.0, i_clamp_min: -10000, i_clamp_max: 10000}

  EffortJointInterface_J2_controller:
    type: effort_controllers/JointPositionController
    joint: j2
    pid: {p: 800.0, i: 10, d: 5.0, i_clamp_min: -10000, i_clamp_max: 10000}

  EffortJointInterface_J3_controller:
    type: effort_controllers/JointPositionController
    joint: j3
    pid: {p: 800.0, i: 10, d: 80.0, i_clamp_min: -10000, i_clamp_max: 10000}


  # Forward Position Controllers --------------------------------------
  PositionJointInterface_J0_controller:
    type: position_controllers/JointPositionController
```



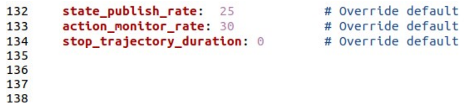
```yaml
    type: position_controllers/JointPositionController
    joint: j0

  PositionJointInterface_J1_controller:
    type: position_controllers/JointPositionController
    joint: j1

  PositionJointInterface_J2_controller:
    type: position_controllers/JointPositionController
    joint: j2

  PositionJointInterface_J3_controller:
    type: position_controllers/JointPositionController
    joint: j3



  # Forward Velocity Controllers --------------------------------------
  VelocityJointInterface_J0_controller:
    type: velocity_controllers/JointVelocityController
    joint: j0

  VelocityJointInterface_J1_controller:
    type: velocity_controllers/JointVelocityController
    joint: j1

  VelocityJointInterface_J2_controller:
    type: velocity_controllers/JointVelocityController
    joint: j2

  VelocityJointInterface_J3_controller:
    type: velocity_controllers/JointVelocityController
    joint: j3

```



```yaml


# Trajectory Controllers --------------------------------------
#
# Effort Position Controllers --------------------------------------
EffortJointInterface_trajectory_controller:
  type: effort_controllers/JointTrajectoryController
  joints:
    - j0
    - j1
    - j2
    - j3

    # VALUES ARE NOT CORRECT !
  gains:
    j0: {p: 500,  d: 30, i: 15, i_clamp: 30}
    j1: {p: 200,  d: 10, i: 10, i_clamp: 30}
    j2: {p: 65,  d: 10, i: 15, i_clamp: 30}
    j3: {p: 31,  d: 7, i: 12, i_clamp: 30}

  constraints:
    goal_time: 0.5                   # Override default

  state_publish_rate:  25            # Override default
  action_monitor_rate: 30            # Override default
  stop_trajectory_duration: 0        # Override default

# Forward Position Controllers --------------------------------------
PositionJointInterface_trajectory_controller:
  type: position_controllers/JointTrajectoryController
  joints:
    - j0
    - j1
```



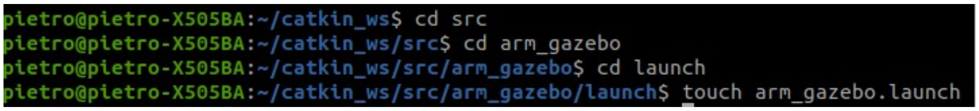
```
      - j1
      - j2
      - j3

    constraints:
      goal_time: 0.5                   # Override default

    state_publish_rate:  25            # Override default
    action_monitor_rate: 30            # Override default
    stop_trajectory_duration: 0        # Override default

  # Forward Velocity Controllers ---------------------------------------
  VelocityJointInterface_trajectory_controller:
    type: velocity_controllers/JointTrajectoryController
    joints:
      - j0
      - j1
      - j2
      - j3

      # VALUES ARE NOT CORRECT !
    gains:
      j0: {p: 500,  d: 30, i: 15, i_clamp: 30}
      j1: {p: 200,  d: 10, i: 10, i_clamp: 30}
      j2: {p: 65,  d: 10, i: 15, i_clamp: 30}
      j3: {p: 31,  d: 7, i: 12, i_clamp: 30}


    constraints:
      goal_time: 0.5                   # Override default

    state_publish_rate:  25            # Override default
    action_monitor_rate: 30            # Override default
```



```
    state_publish_rate:  25            # Override default
    action_monitor_rate: 30            # Override default
    stop_trajectory_duration: 0        # Override default
```

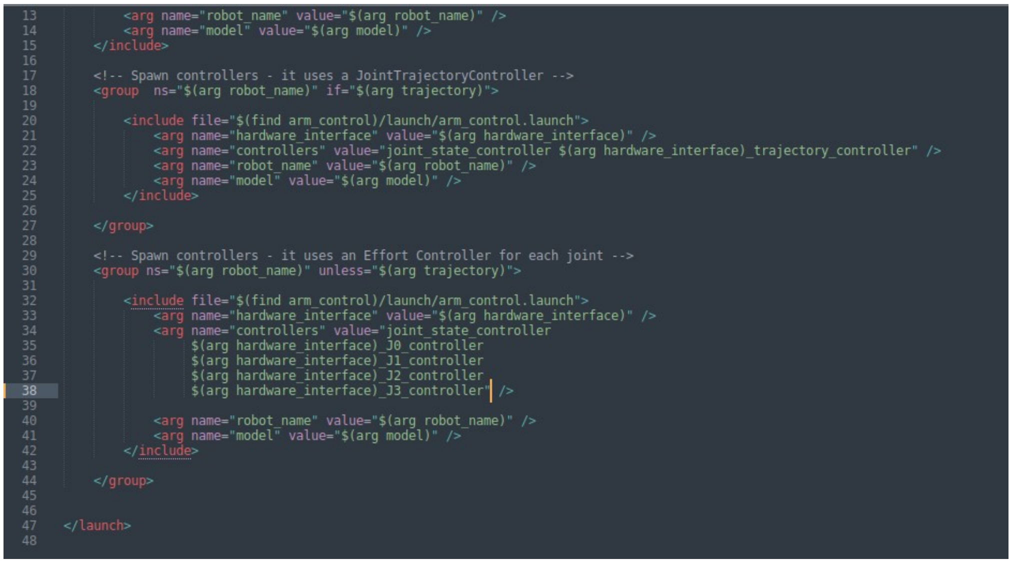
2.g Create an *arm_gazebo.launch* file into the *launch* folder of the *arm_gazebo* package loading the Gazebo world with *arm_world.launch* and spawning the controllers within *arm_control.launch*. Go to the *arm_description* package and add the *gazebo_ros_control* plugin to your main URDF into the *arm.gazebo.xacro* file. Launch the simulation and check if your controllers are correctly loaded

**terminal:**



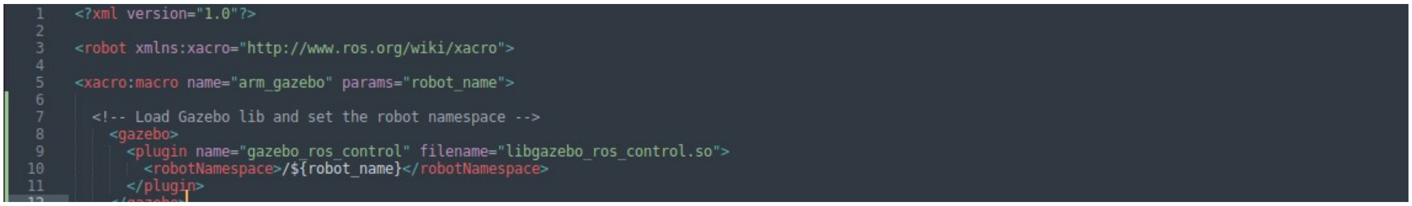
```
pietro@pietro-X505BA:~/catkin_ws$ cd src
pietro@pietro-X505BA:~/catkin_ws/src$ cd arm_gazebo
pietro@pietro-X505BA:~/catkin_ws/src/arm_gazebo$ cd launch
pietro@pietro-X505BA:~/catkin_ws/src/arm_gazebo/launch$ touch arm_gazebo.launch
```

**arm_gazebo.launch:**



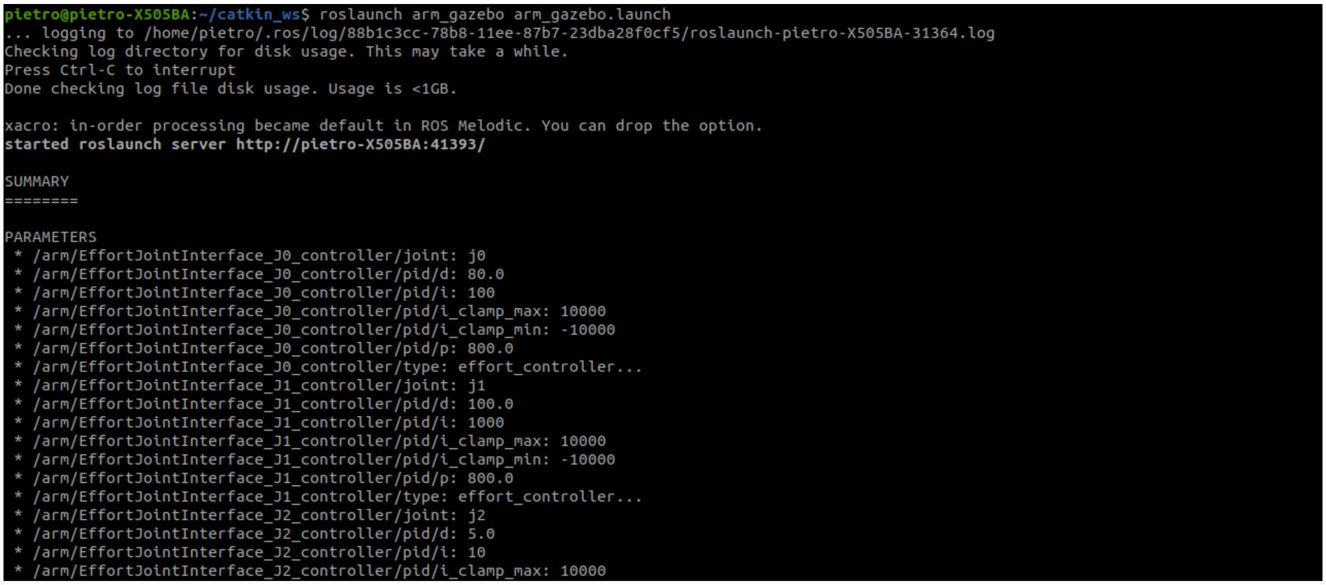
```
            <arg name="robot_name" value="$(arg robot_name)" />
            <arg name="model" value="$(arg model)" />
        </include>

        <!-- Spawn controllers - it uses a JointTrajectoryController -->
        <group  ns="$(arg robot_name)" if="$(arg trajectory)">

            <include file="$(find arm_control)/launch/arm_control.launch">
                <arg name="hardware_interface" value="$(arg hardware_interface)" />
                <arg name="controllers" value="joint_state_controller $(arg hardware_interface)_trajectory_controller" />
                <arg name="robot_name" value="$(arg robot_name)" />
                <arg name="model" value="$(arg model)" />
            </include>

        </group>

        <!-- Spawn controllers - it uses an Effort Controller for each joint -->
        <group ns="$(arg robot_name)" unless="$(arg trajectory)">

            <include file="$(find arm_control)/launch/arm_control.launch">
                <arg name="hardware_interface" value="$(arg hardware_interface)" />
                <arg name="controllers" value="joint_state_controller
                     $(arg hardware_interface)_J0_controller
                     $(arg hardware_interface)_J1_controller
                     $(arg hardware_interface)_J2_controller
                     $(arg hardware_interface)_J3_controller" />

                <arg name="robot_name" value="$(arg robot_name)" />
                <arg name="model" value="$(arg model)" />
            </include>

        </group>


    </launch>
```

**arm_gazebo.xacro:** I added "gazebo_ros_control" plugin

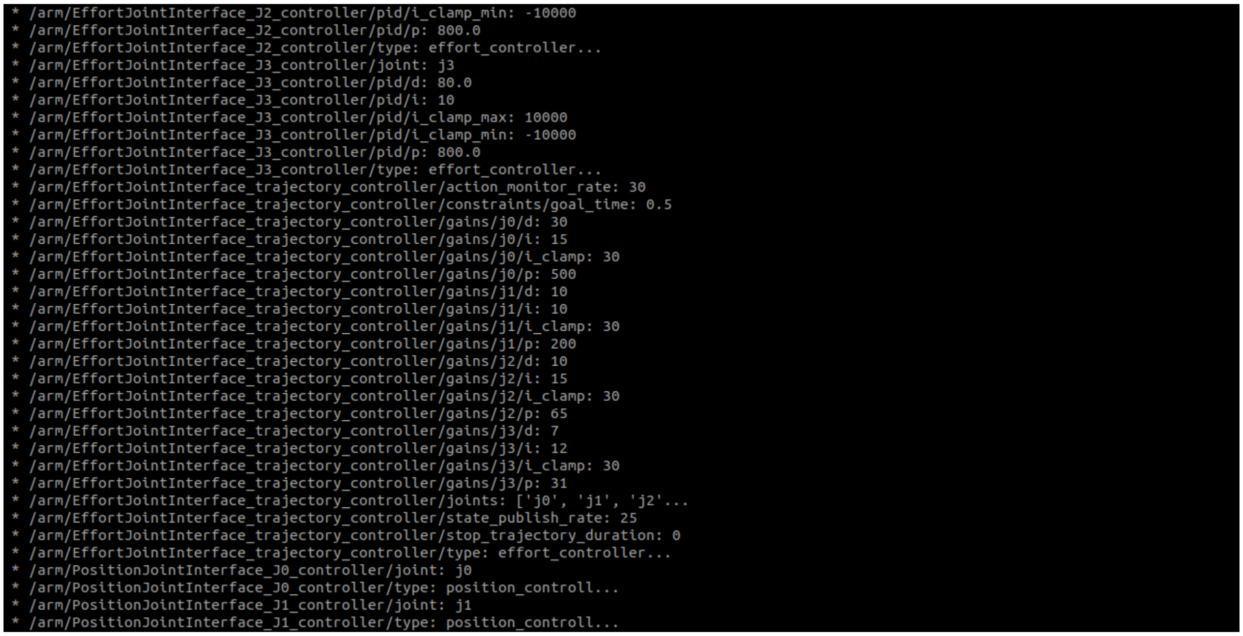


```
<?xml version="1.0"?>

<robot xmlns:xacro="http://www.ros.org/wiki/xacro">

<xacro:macro name="arm_gazebo" params="robot_name">

  <!-- Load Gazebo lib and set the robot namespace -->
    <gazebo>
      <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
        <robotNamespace>/${robot_name}</robotNamespace>
      </plugin>
```

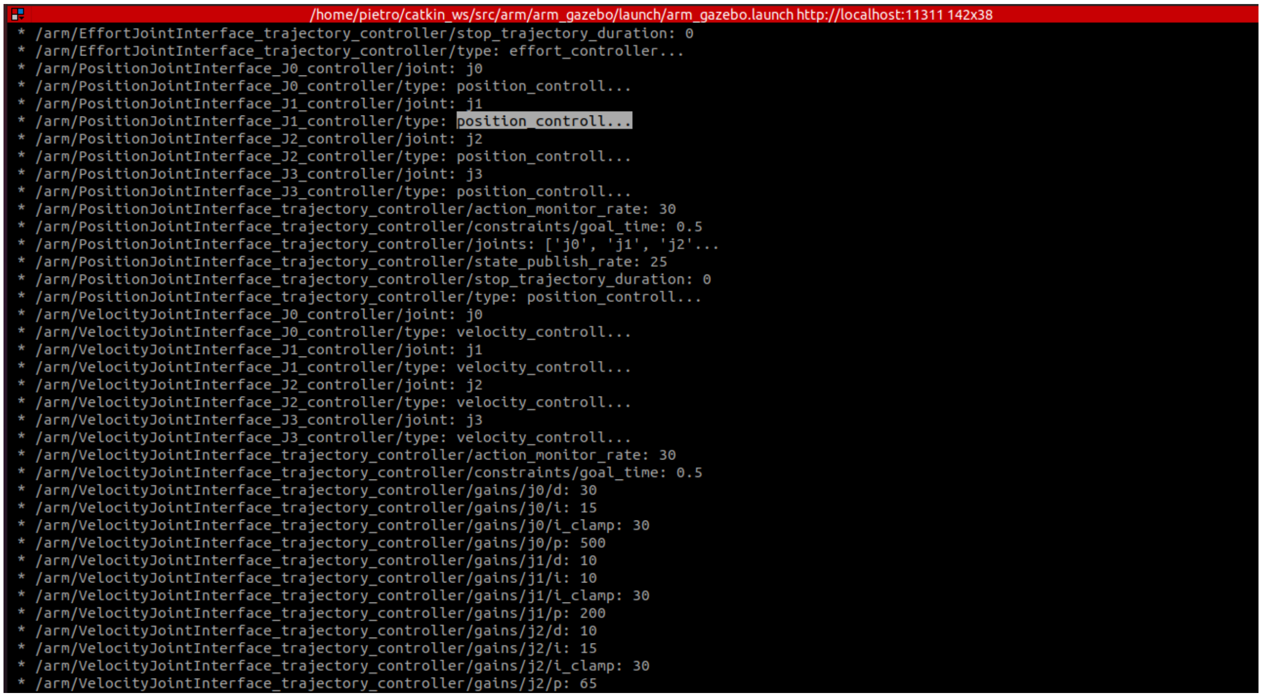
**check:** “roslaunch arm_gazebo arm_gazebo.launch” **by terminal**



```
pietro@pietro-X505BA:~/catkin_ws$ roslaunch arm_gazebo arm_gazebo.launch
... logging to /home/pietro/.ros/log/88b1c3cc-78b8-11ee-87b7-23dba28f0cf5/roslaunch-pietro-X505BA-31364.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: in-order processing became default in ROS Melodic. You can drop the option.
started roslaunch server http://pietro-X505BA:41393/

SUMMARY
========

PARAMETERS
 * /arm/EffortJointInterface_J0_controller/joint: j0
 * /arm/EffortJointInterface_J0_controller/pid/d: 80.0
 * /arm/EffortJointInterface_J0_controller/pid/i: 100
 * /arm/EffortJointInterface_J0_controller/pid/i_clamp_max: 10000
 * /arm/EffortJointInterface_J0_controller/pid/i_clamp_min: -10000
 * /arm/EffortJointInterface_J0_controller/pid/p: 800.0
 * /arm/EffortJointInterface_J0_controller/type: effort_controller...
 * /arm/EffortJointInterface_J1_controller/joint: j1
 * /arm/EffortJointInterface_J1_controller/pid/d: 100.0
 * /arm/EffortJointInterface_J1_controller/pid/i: 1000
 * /arm/EffortJointInterface_J1_controller/pid/i_clamp_max: 10000
 * /arm/EffortJointInterface_J1_controller/pid/i_clamp_min: -10000
 * /arm/EffortJointInterface_J1_controller/pid/p: 800.0
 * /arm/EffortJointInterface_J1_controller/type: effort_controller...
 * /arm/EffortJointInterface_J2_controller/joint: j2
 * /arm/EffortJointInterface_J2_controller/pid/d: 5.0
 * /arm/EffortJointInterface_J2_controller/pid/i: 10
 * /arm/EffortJointInterface_J2_controller/pid/i_clamp_max: 10000
```

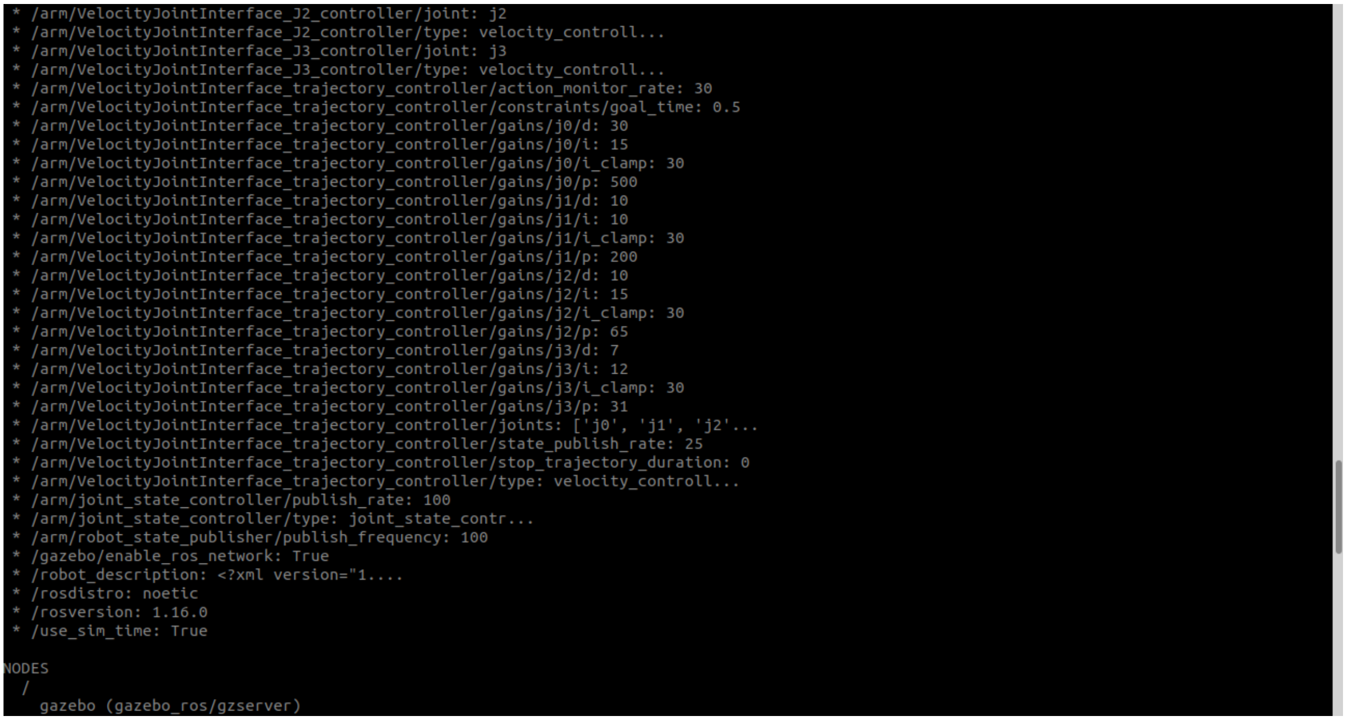


```
 * /arm/EffortJointInterface_J2_controller/pid/i_clamp_min: -10000
 * /arm/EffortJointInterface_J2_controller/pid/p: 800.0
 * /arm/EffortJointInterface_J2_controller/type: effort_controller...
 * /arm/EffortJointInterface_J3_controller/joint: j3
 * /arm/EffortJointInterface_J3_controller/pid/d: 80.0
 * /arm/EffortJointInterface_J3_controller/pid/i: 10
 * /arm/EffortJointInterface_J3_controller/pid/i_clamp_max: 10000
 * /arm/EffortJointInterface_J3_controller/pid/i_clamp_min: -10000
 * /arm/EffortJointInterface_J3_controller/pid/p: 800.0
 * /arm/EffortJointInterface_J3_controller/type: effort_controller...
 * /arm/EffortJointInterface_trajectory_controller/action_monitor_rate: 30
 * /arm/EffortJointInterface_trajectory_controller/constraints/goal_time: 0.5
 * /arm/EffortJointInterface_trajectory_controller/gains/j0/d: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j0/i: 15
 * /arm/EffortJointInterface_trajectory_controller/gains/j0/i_clamp: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j0/p: 500
 * /arm/EffortJointInterface_trajectory_controller/gains/j1/d: 10
 * /arm/EffortJointInterface_trajectory_controller/gains/j1/i: 10
 * /arm/EffortJointInterface_trajectory_controller/gains/j1/i_clamp: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j1/p: 200
 * /arm/EffortJointInterface_trajectory_controller/gains/j2/d: 10
 * /arm/EffortJointInterface_trajectory_controller/gains/j2/i: 15
 * /arm/EffortJointInterface_trajectory_controller/gains/j2/i_clamp: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j2/p: 65
 * /arm/EffortJointInterface_trajectory_controller/gains/j3/d: 7
 * /arm/EffortJointInterface_trajectory_controller/gains/j3/i: 12
 * /arm/EffortJointInterface_trajectory_controller/gains/j3/i_clamp: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j3/p: 31
 * /arm/EffortJointInterface_trajectory_controller/joints: ['j0', 'j1', 'j2'...
 * /arm/EffortJointInterface_trajectory_controller/state_publish_rate: 25
 * /arm/EffortJointInterface_trajectory_controller/stop_trajectory_duration: 0
 * /arm/EffortJointInterface_trajectory_controller/type: effort_controller...
 * /arm/PositionJointInterface_J0_controller/joint: j0
 * /arm/PositionJointInterface_J0_controller/type: position_controll...
 * /arm/PositionJointInterface_J1_controller/joint: j1
 * /arm/PositionJointInterface_J1_controller/type: position_controll...
```

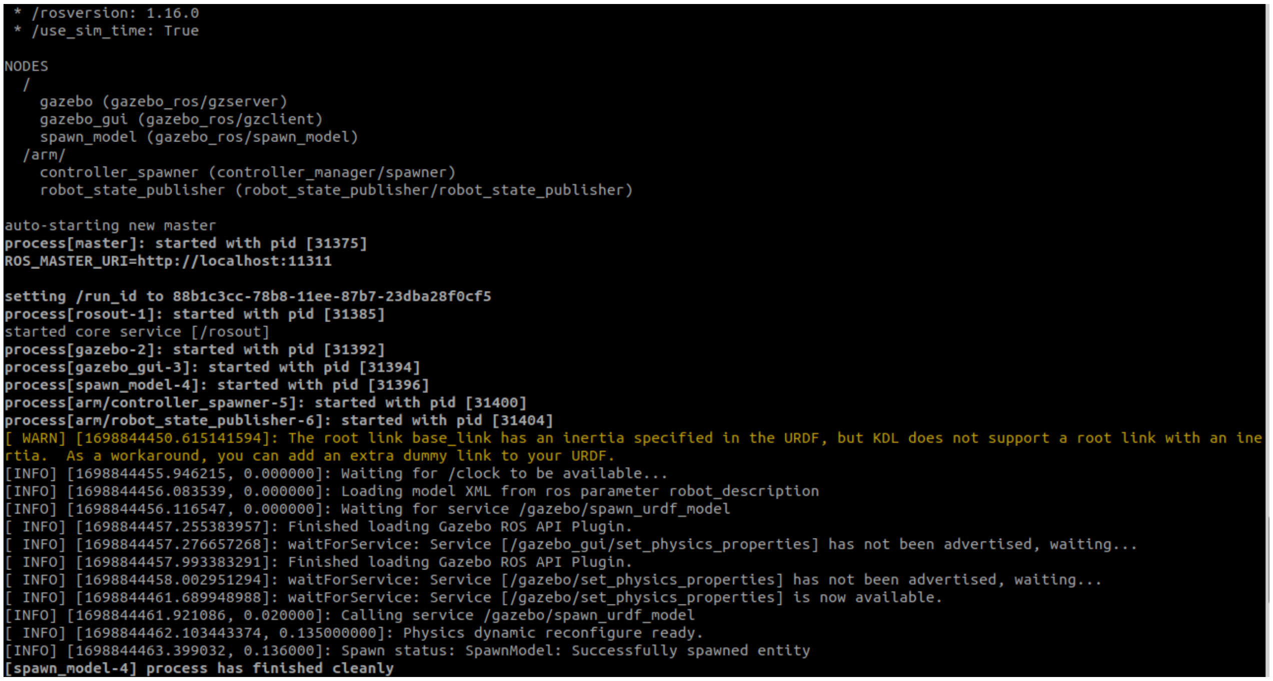


/home/pietro/catkin_ws/src/arm/arm_gazebo/launch/arm_gazebo.launch http://localhost:11311 142x38

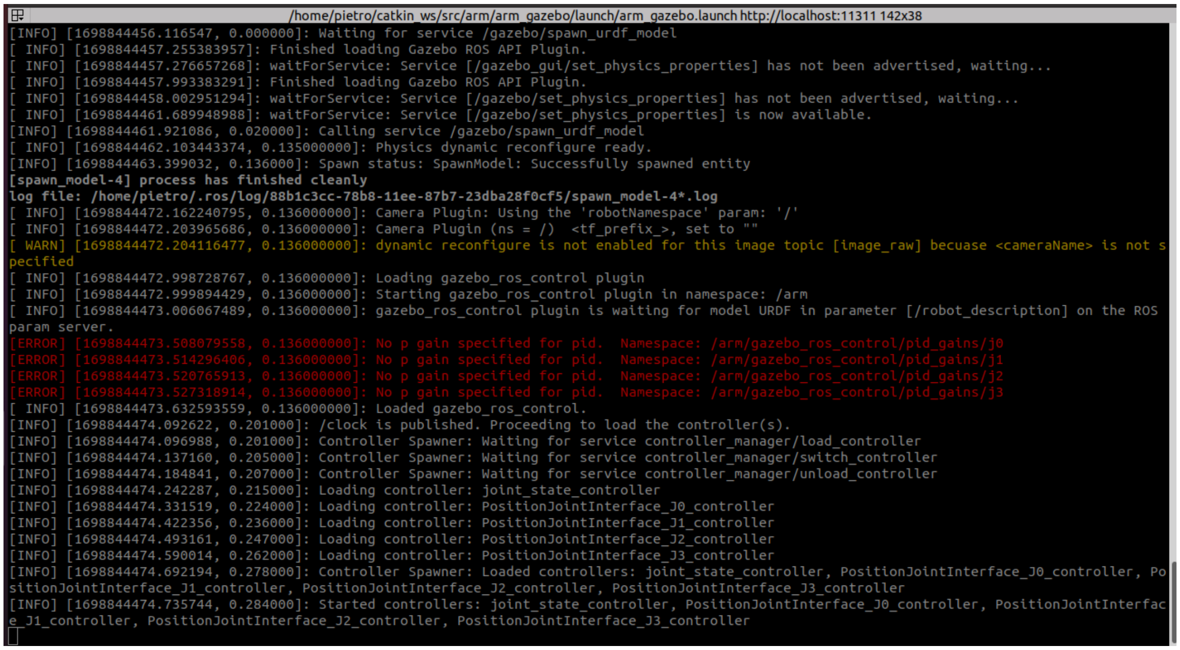
```
 * /arm/EffortJointInterface_trajectory_controller/stop_trajectory_duration: 0
 * /arm/EffortJointInterface_trajectory_controller/type: effort_controller...
 * /arm/PositionJointInterface_J0_controller/joint: j0
 * /arm/PositionJointInterface_J0_controller/type: position_controll...
 * /arm/PositionJointInterface_J1_controller/joint: j1
 * /arm/PositionJointInterface_J1_controller/type: position_controll...
 * /arm/PositionJointInterface_J2_controller/joint: j2
 * /arm/PositionJointInterface_J2_controller/type: position_controll...
 * /arm/PositionJointInterface_J3_controller/joint: j3
 * /arm/PositionJointInterface_J3_controller/type: position_controll...
 * /arm/PositionJointInterface_trajectory_controller/action_monitor_rate: 30
 * /arm/PositionJointInterface_trajectory_controller/constraints/goal_time: 0.5
 * /arm/PositionJointInterface_trajectory_controller/joints: ['j0', 'j1', 'j2'...
 * /arm/PositionJointInterface_trajectory_controller/state_publish_rate: 25
 * /arm/PositionJointInterface_trajectory_controller/stop_trajectory_duration: 0
 * /arm/PositionJointInterface_trajectory_controller/type: position_controll...
 * /arm/VelocityJointInterface_J0_controller/joint: j0
 * /arm/VelocityJointInterface_J0_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_J1_controller/joint: j1
 * /arm/VelocityJointInterface_J1_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_J2_controller/joint: j2
 * /arm/VelocityJointInterface_J2_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_J3_controller/joint: j3
 * /arm/VelocityJointInterface_J3_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_trajectory_controller/action_monitor_rate: 30
 * /arm/VelocityJointInterface_trajectory_controller/constraints/goal_time: 0.5
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/d: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/i: 15
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/p: 500
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/d: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/i: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/p: 200
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/d: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/i: 15
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/p: 65
```



```
 * /arm/VelocityJointInterface_J2_controller/joint: j2
 * /arm/VelocityJointInterface_J2_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_J3_controller/joint: j3
 * /arm/VelocityJointInterface_J3_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_trajectory_controller/action_monitor_rate: 30
 * /arm/VelocityJointInterface_trajectory_controller/constraints/goal_time: 0.5
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/d: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/i: 15
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/p: 500
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/d: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/i: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/p: 200
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/d: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/i: 15
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/p: 65
 * /arm/VelocityJointInterface_trajectory_controller/gains/j3/d: 7
 * /arm/VelocityJointInterface_trajectory_controller/gains/j3/i: 12
 * /arm/VelocityJointInterface_trajectory_controller/gains/j3/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j3/p: 31
 * /arm/VelocityJointInterface_trajectory_controller/joints: ['j0', 'j1', 'j2'...
 * /arm/VelocityJointInterface_trajectory_controller/state_publish_rate: 25
 * /arm/VelocityJointInterface_trajectory_controller/stop_trajectory_duration: 0
 * /arm/VelocityJointInterface_trajectory_controller/type: velocity_controll...
 * /arm/joint_state_controller/publish_rate: 100
 * /arm/joint_state_controller/type: joint_state_contr...
 * /arm/robot_state_publisher/publish_frequency: 100
 * /gazebo/enable_ros_network: True
 * /robot_description: <?xml version="1....
 * /rosdistro: noetic
 * /rosversion: 1.16.0
 * /use_sim_time: True

NODES
  /
    gazebo (gazebo_ros/gzserver)
```

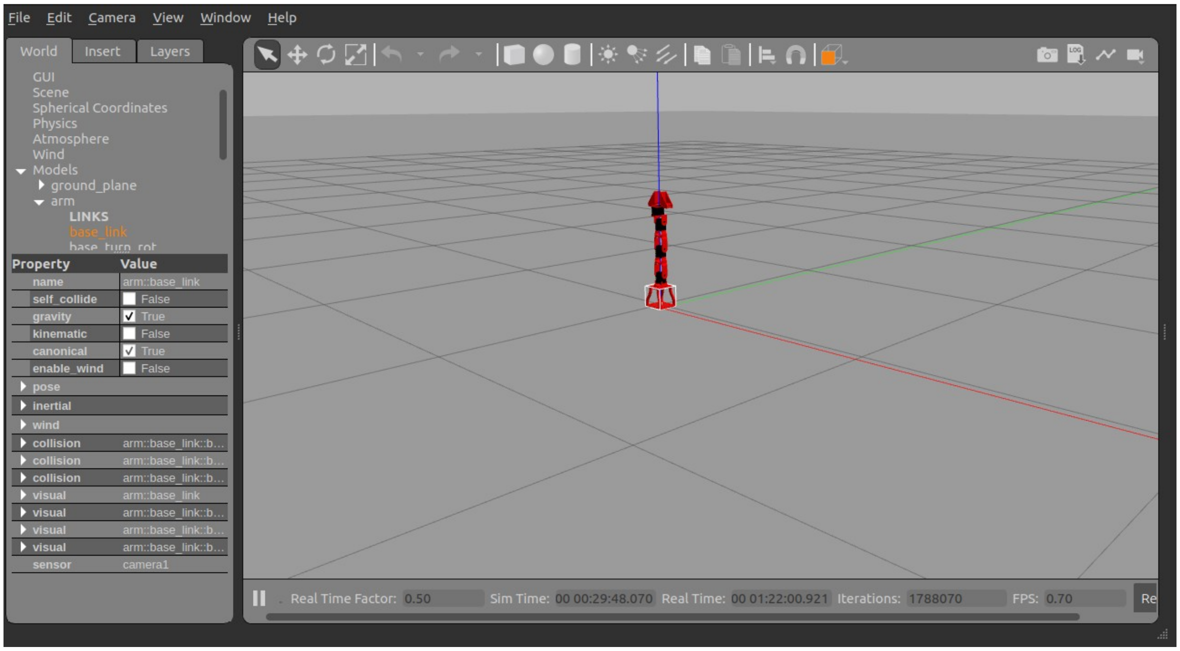


```
 * /rosversion: 1.16.0
 * /use_sim_time: True

NODES
  /
    gazebo (gazebo_ros/gzserver)
    gazebo_gui (gazebo_ros/gzclient)
    spawn_model (gazebo_ros/spawn_model)
  /arm/
    controller_spawner (controller_manager/spawner)
    robot_state_publisher (robot_state_publisher/robot_state_publisher)

auto-starting new master
process[master]: started with pid [31375]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 88b1c3cc-78b8-11ee-87b7-23dba28f0cf5
process[rosout-1]: started with pid [31385]
started core service [/rosout]
process[gazebo-2]: started with pid [31392]
process[gazebo_gui-3]: started with pid [31394]
process[spawn_model-4]: started with pid [31396]
process[arm/controller_spawner-5]: started with pid [31400]
process[arm/robot_state_publisher-6]: started with pid [31404]
[ WARN] [1698844450.615141594]: The root link base_link has an inertia specified in the URDF, but KDL does not support a root link with an ine
rtia.  As a workaround, you can add an extra dummy link to your URDF.
[INFO] [1698844455.946215, 0.000000]: Waiting for /clock to be available...
[INFO] [1698844456.083539, 0.000000]: Loading model XML from ros parameter robot_description
[INFO] [1698844456.116547, 0.000000]: Waiting for service /gazebo/spawn_urdf_model
[ INFO] [1698844457.255383957]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1698844457.276657268]: waitForService: Service [/gazebo_gui/set_physics_properties] has not been advertised, waiting...
[ INFO] [1698844457.993383291]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1698844458.002951294]: waitForService: Service [/gazebo/set_physics_properties] has not been advertised, waiting...
[ INFO] [1698844461.689948988]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1698844461.921086, 0.020000]: Calling service /gazebo/spawn_urdf_model
[ INFO] [1698844462.103443374, 0.135000000]: Physics dynamic reconfigure ready.
[INFO] [1698844463.399032, 0.136000]: Spawn status: SpawnModel: Successfully spawned entity
[spawn_model-4] process has finished cleanly
```



/home/pietro/catkin_ws/src/arm/arm_gazebo/launch/arm_gazebo.launch http://localhost:11311 142x38

```
[INFO] [1698844456.116547, 0.000000]: Waiting for service /gazebo/spawn_urdf_model
[ INFO] [1698844457.255383957]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1698844457.276657268]: waitForService: Service [/gazebo_gui/set_physics_properties] has not been advertised, waiting...
[ INFO] [1698844457.993383291]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1698844458.002951294]: waitForService: Service [/gazebo/set_physics_properties] has not been advertised, waiting...
[ INFO] [1698844461.689948988]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1698844461.921086, 0.020000]: Calling service /gazebo/spawn_urdf_model
[ INFO] [1698844462.103443374, 0.135000000]: Physics dynamic reconfigure ready.
[INFO] [1698844463.399032, 0.136000]: Spawn status: SpawnModel: Successfully spawned entity
[spawn_model-4] process has finished cleanly
log file: /home/pietro/.ros/log/88b1c3cc-78b8-11ee-87b7-23dba28f0cf5/spawn_model-4*.log
[ INFO] [1698844472.162240795, 0.136000000]: Camera Plugin: Using the 'robotNamespace' param: '/'
[ INFO] [1698844472.203965686, 0.136000000]: Camera Plugin (ns = /)  <tf_prefix_>, set to ""
[ WARN] [1698844472.204116477, 0.136000000]: dynamic reconfigure is not enabled for this image topic [image_raw] becuase <cameraName> is not s
pecified
[ INFO] [1698844472.998728767, 0.136000000]: Loading gazebo_ros_control plugin
[ INFO] [1698844472.999894429, 0.136000000]: Starting gazebo_ros_control plugin in namespace: /arm
[ INFO] [1698844473.006067489, 0.136000000]: gazebo_ros_control plugin is waiting for model URDF in parameter [/robot_description] on the ROS
param server.
[ERROR] [1698844473.508079558, 0.136000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j0
[ERROR] [1698844473.514296406, 0.136000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j1
[ERROR] [1698844473.520765913, 0.136000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j2
[ERROR] [1698844473.527318914, 0.136000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j3
[ INFO] [1698844473.632593559, 0.136000000]: Loaded gazebo_ros_control.
[INFO] [1698844474.092622, 0.201000]: /clock is published. Proceeding to load the controller(s).
[INFO] [1698844474.096988, 0.201000]: Controller Spawner: Waiting for service controller_manager/load_controller
[INFO] [1698844474.137160, 0.205000]: Controller Spawner: Waiting for service controller_manager/switch_controller
[INFO] [1698844474.184841, 0.207000]: Controller Spawner: Waiting for service controller_manager/unload_controller
[INFO] [1698844474.242287, 0.215000]: Loading controller: joint_state_controller
[INFO] [1698844474.331519, 0.224000]: Loading controller: PositionJointInterface_J0_controller
[INFO] [1698844474.422356, 0.236000]: Loading controller: PositionJointInterface_J1_controller
[INFO] [1698844474.493161, 0.247000]: Loading controller: PositionJointInterface_J2_controller
[INFO] [1698844474.590014, 0.262000]: Loading controller: PositionJointInterface_J3_controller
[INFO] [1698844474.692194, 0.278000]: Controller Spawner: Loaded controllers: joint_state_controller, PositionJointInterface_J0_controller, Po
sitionJointInterface_J1_controller, PositionJointInterface_J2_controller, PositionJointInterface_J3_controller
[INFO] [1698844474.735744, 0.284000]: Started controllers: joint_state_controller, PositionJointInterface_J0_controller, PositionJointInterfac
e_J1_controller, PositionJointInterface_J2_controller, PositionJointInterface_J3_controller
```



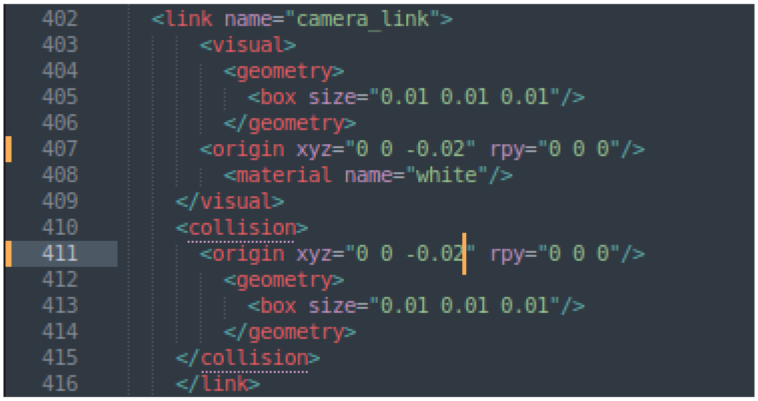
| Property | Value |
|---|---|
| name | arm::base_link |
| self_collide | □ False |
| gravity | ☑ True |
| kinematic | □ False |
| canonical | ☑ True |
| enable_wind | □ False |
| pose | |
| inertial | |
| wind | |
| collision | arm::base_link::b... |
| collision | arm::base_link::b... |
| collision | arm::base_link::b... |
| visual | arm::base_link |
| visual | arm::base_link::b... |
| visual | arm::base_link::b... |
| visual | arm::base_link::b... |
| sensor | camera1 |

Real Time Factor: 0.50 Sim Time: 00 00:29:48.070 Real Time: 00 01:22:00.921 Iterations: 1788070 FPS: 0.70

## 3 Add a camera sensor to your robot

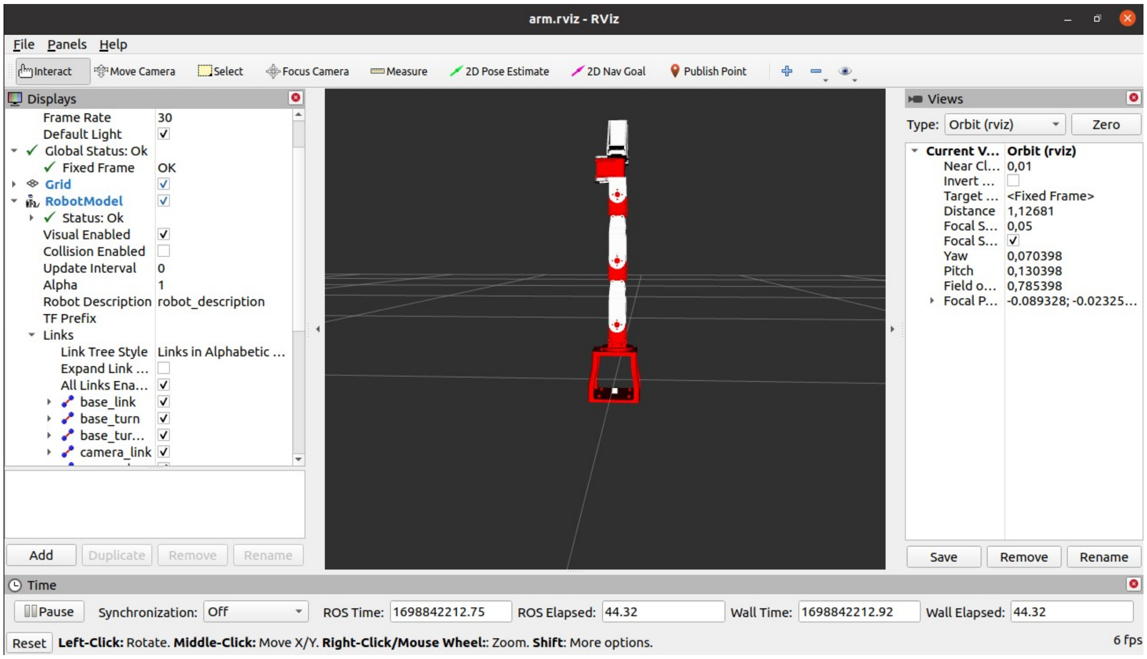
3.a Go into your *arm.urdf.xacro* file and add a *camera_link* and a fixed *camera_joint* with *base_link* as a parent link. Size and position the camera link opportunely

**arm.urdf.xacro:** I added this lines to define the camera_link and to add the camera_link to the robot and I defined the fixed camera_joint



```
<link name="camera_link">
  <visual>
    <geometry>
      <box size="0.01 0.01 0.01"/>
    </geometry>
    <origin xyz="0 0 -0.02" rpy="0 0 0"/>
      <material name="white"/>
  </visual>
  <collision>
    <origin xyz="0 0 -0.02" rpy="0 0 0"/>
    <geometry>
      <box size="0.01 0.01 0.01"/>
    </geometry>
  </collision>
</link>
```

**check Rrviz:** I positioned the camera_link to the base_link and I gave a box size "0.01 0.01 0.01"and origin regard xyz= "0 0 -0.02" so to pose it on the base_link. I disabled collissions to view the camera_link.



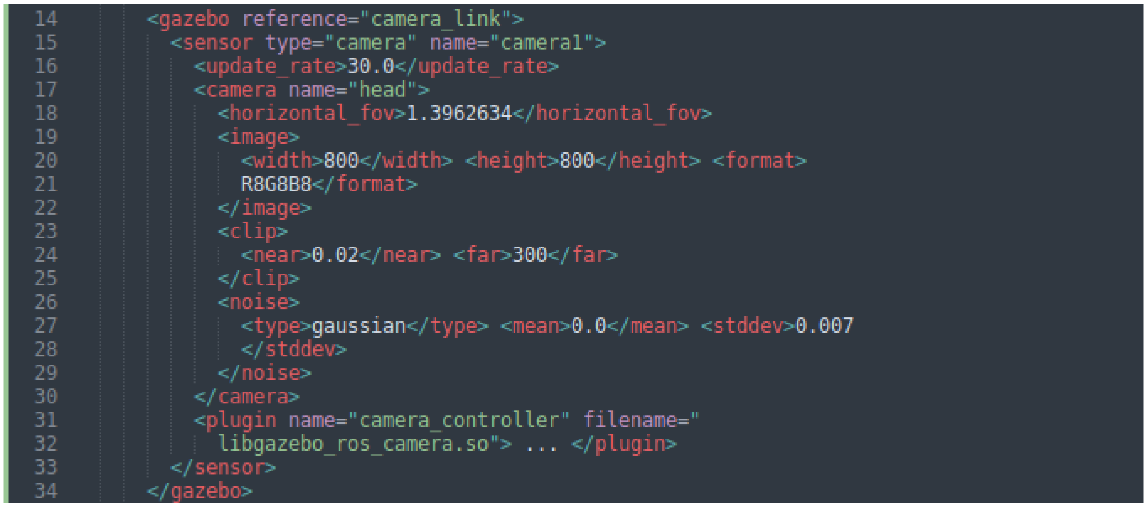
```
Frame Rate            30
Default Light         ☑
Global Status: Ok
  Fixed Frame         OK
Grid                  ☑
RobotModel            ☑
  Status: Ok
  Visual Enabled      ☑
  Collision Enabled   □
  Update Interval     0
  Alpha               1
  Robot Description   robot_description
  TF Prefix
  Links
    Link Tree Style   Links in Alphabetic ...
    Expand Link ...   □
    All Links Ena...  ☑
    base_link         ☑
    base_turn         ☑
    base_tur...       ☑
    camera_link       ☑
```

```
Type: Orbit (rviz)
Current V...   Orbit (rviz)
  Near Cl...   0,01
  Invert ...   □
  Target ...   <Fixed Frame>
  Distance     1,12681
  Focal S...   0,05
  Focal S...   ☑
  Yaw          0,070398
  Pitch        0,130398
  Field o...   0,785398
  Focal P...   -0.089328; -0.02325...
```

```
Synchronization: Off   ROS Time: 1698842212.75   ROS Elapsed: 44.32   Wall Time: 1698842212.92   Wall Elapsed: 44.32
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.   6 fps
```

3.b In the *arm.gazebo.xacro* add the gazebo sensor reference tags and the *libgazebo_ros_camera* plugin to your xacro (slide 74-75)

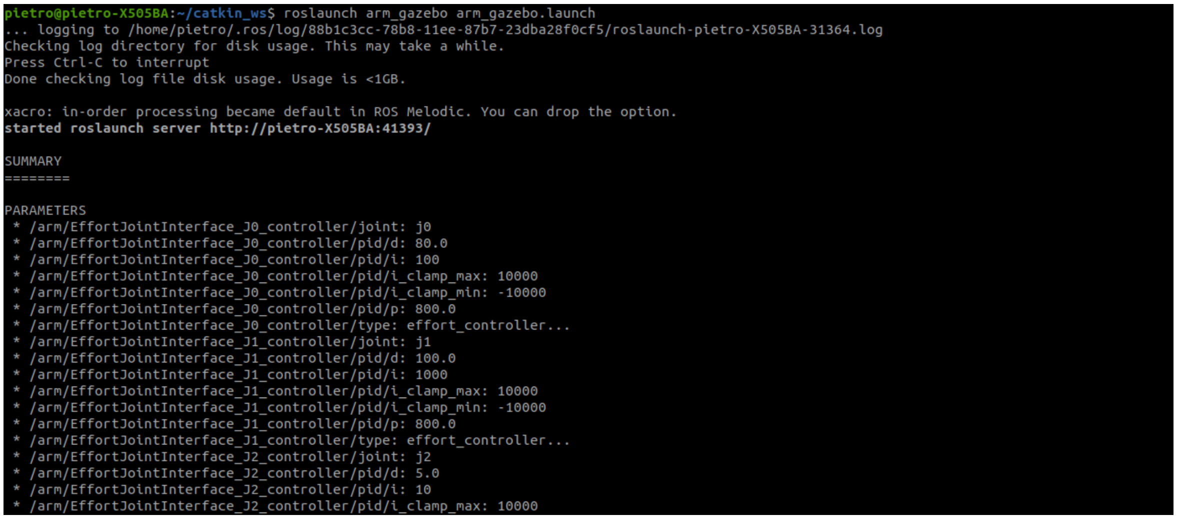
**arm.gazebo.xacro:**



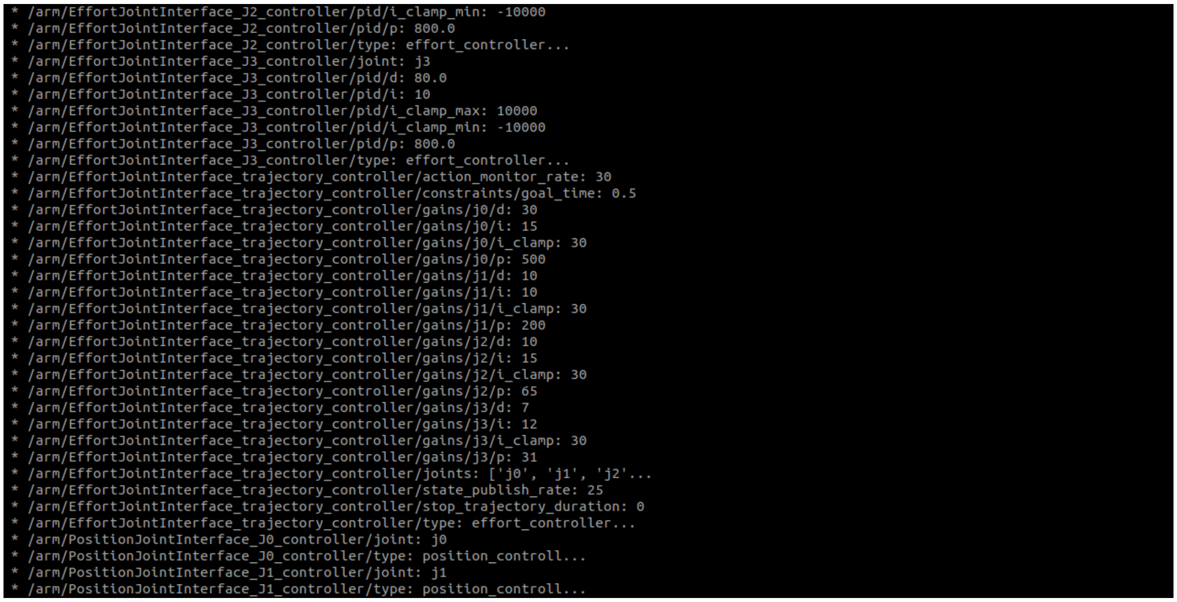
```
    <gazebo reference="camera_link">
      <sensor type="camera" name="camera1">
        <update_rate>30.0</update_rate>
        <camera name="head">
          <horizontal_fov>1.3962634</horizontal_fov>
          <image>
            <width>800</width> <height>800</height> <format>
            R8G8B8</format>
          </image>
          <clip>
            <near>0.02</near> <far>300</far>
          </clip>
          <noise>
            <type>gaussian</type> <mean>0.0</mean> <stddev>0.007
            </stddev>
          </noise>
        </camera>
        <plugin name="camera_controller" filename="
          libgazebo_ros_camera.so"> ... </plugin>
      </sensor>
    </gazebo>
```

3.c Launch the Gazebo simulation with using *arm_gazebo.launch* and check if the image topic is correctly published using *rqt_image_view*

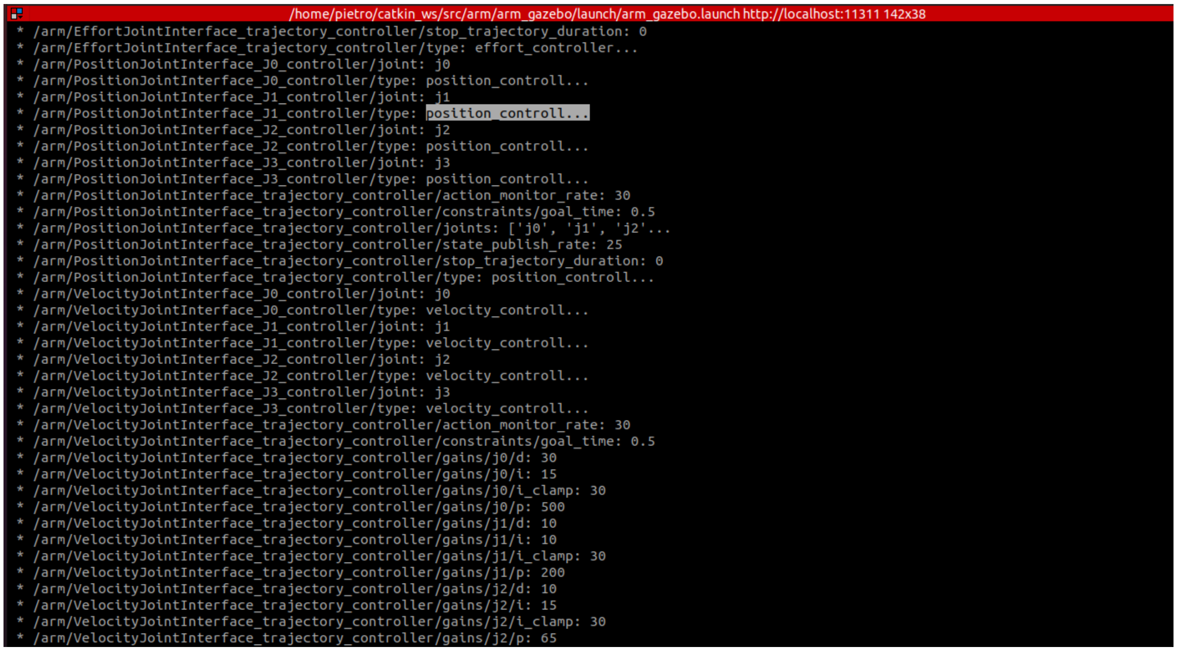
*"roslaunch arm_gazebo arm_gazebo.launch"* **by terminal:**



```
pietro@pietro-X505BA:~/catkin_ws$ roslaunch arm_gazebo arm_gazebo.launch
... logging to /home/pietro/.ros/log/88b1c3cc-78b8-11ee-87b7-23dba28f0cf5/roslaunch-pietro-X505BA-31364.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

xacro: in-order processing became default in ROS Melodic. You can drop the option.
started roslaunch server http://pietro-X505BA:41393/

SUMMARY
========

PARAMETERS
 * /arm/EffortJointInterface_J0_controller/joint: j0
 * /arm/EffortJointInterface_J0_controller/pid/d: 80.0
 * /arm/EffortJointInterface_J0_controller/pid/i: 100
 * /arm/EffortJointInterface_J0_controller/pid/i_clamp_max: 10000
 * /arm/EffortJointInterface_J0_controller/pid/i_clamp_min: -10000
 * /arm/EffortJointInterface_J0_controller/pid/p: 800.0
 * /arm/EffortJointInterface_J0_controller/type: effort_controller...
 * /arm/EffortJointInterface_J1_controller/joint: j1
 * /arm/EffortJointInterface_J1_controller/pid/d: 100.0
 * /arm/EffortJointInterface_J1_controller/pid/i: 1000
 * /arm/EffortJointInterface_J1_controller/pid/i_clamp_max: 10000
 * /arm/EffortJointInterface_J1_controller/pid/i_clamp_min: -10000
 * /arm/EffortJointInterface_J1_controller/pid/p: 800.0
 * /arm/EffortJointInterface_J1_controller/type: effort_controller...
 * /arm/EffortJointInterface_J2_controller/joint: j2
 * /arm/EffortJointInterface_J2_controller/pid/d: 5.0
 * /arm/EffortJointInterface_J2_controller/pid/i: 10
 * /arm/EffortJointInterface_J2_controller/pid/i_clamp_max: 10000
```

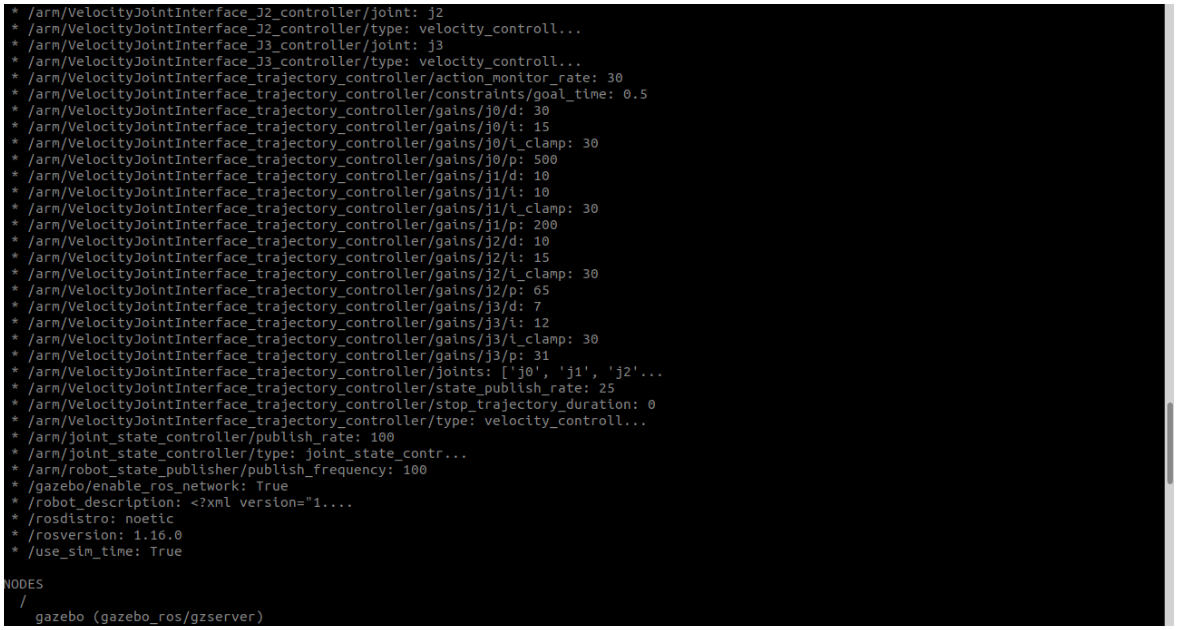


```
 * /arm/EffortJointInterface_J2_controller/pid/i_clamp_min: -10000
 * /arm/EffortJointInterface_J2_controller/pid/p: 800.0
 * /arm/EffortJointInterface_J2_controller/type: effort_controller...
 * /arm/EffortJointInterface_J3_controller/joint: j3
 * /arm/EffortJointInterface_J3_controller/pid/d: 80.0
 * /arm/EffortJointInterface_J3_controller/pid/i: 10
 * /arm/EffortJointInterface_J3_controller/pid/i_clamp_max: 10000
 * /arm/EffortJointInterface_J3_controller/pid/i_clamp_min: -10000
 * /arm/EffortJointInterface_J3_controller/pid/p: 800.0
 * /arm/EffortJointInterface_J3_controller/type: effort_controller...
 * /arm/EffortJointInterface_trajectory_controller/action_monitor_rate: 30
 * /arm/EffortJointInterface_trajectory_controller/constraints/goal_time: 0.5
 * /arm/EffortJointInterface_trajectory_controller/gains/j0/d: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j0/i: 15
 * /arm/EffortJointInterface_trajectory_controller/gains/j0/i_clamp: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j0/p: 500
 * /arm/EffortJointInterface_trajectory_controller/gains/j1/d: 10
 * /arm/EffortJointInterface_trajectory_controller/gains/j1/i: 10
 * /arm/EffortJointInterface_trajectory_controller/gains/j1/i_clamp: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j1/p: 200
 * /arm/EffortJointInterface_trajectory_controller/gains/j2/d: 10
 * /arm/EffortJointInterface_trajectory_controller/gains/j2/i: 15
 * /arm/EffortJointInterface_trajectory_controller/gains/j2/i_clamp: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j2/p: 65
 * /arm/EffortJointInterface_trajectory_controller/gains/j3/d: 7
 * /arm/EffortJointInterface_trajectory_controller/gains/j3/i: 12
 * /arm/EffortJointInterface_trajectory_controller/gains/j3/i_clamp: 30
 * /arm/EffortJointInterface_trajectory_controller/gains/j3/p: 31
 * /arm/EffortJointInterface_trajectory_controller/joints: ['j0', 'j1', 'j2'...
 * /arm/EffortJointInterface_trajectory_controller/state_publish_rate: 25
 * /arm/EffortJointInterface_trajectory_controller/stop_trajectory_duration: 0
 * /arm/EffortJointInterface_trajectory_controller/type: effort_controller...
 * /arm/PositionJointInterface_J0_controller/joint: j0
 * /arm/PositionJointInterface_J0_controller/type: position_controll...
 * /arm/PositionJointInterface_J1_controller/joint: j1
 * /arm/PositionJointInterface_J1_controller/type: position_controll...
```

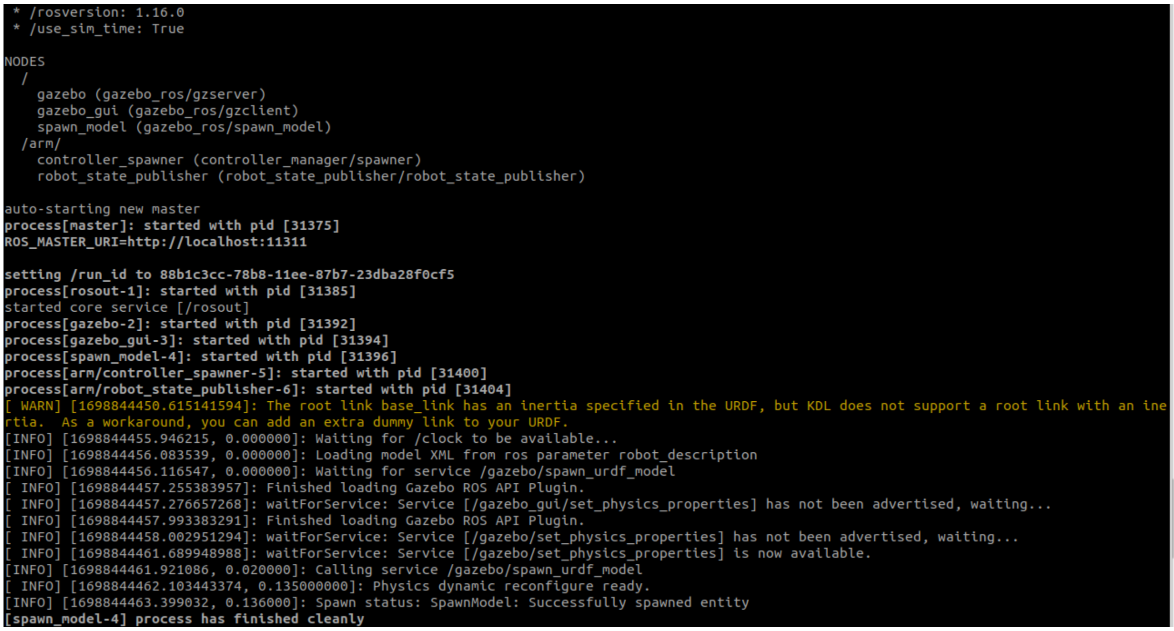


/home/pietro/catkin_ws/src/arm/arm_gazebo/launch/arm_gazebo.launch http://localhost:11311 142x38

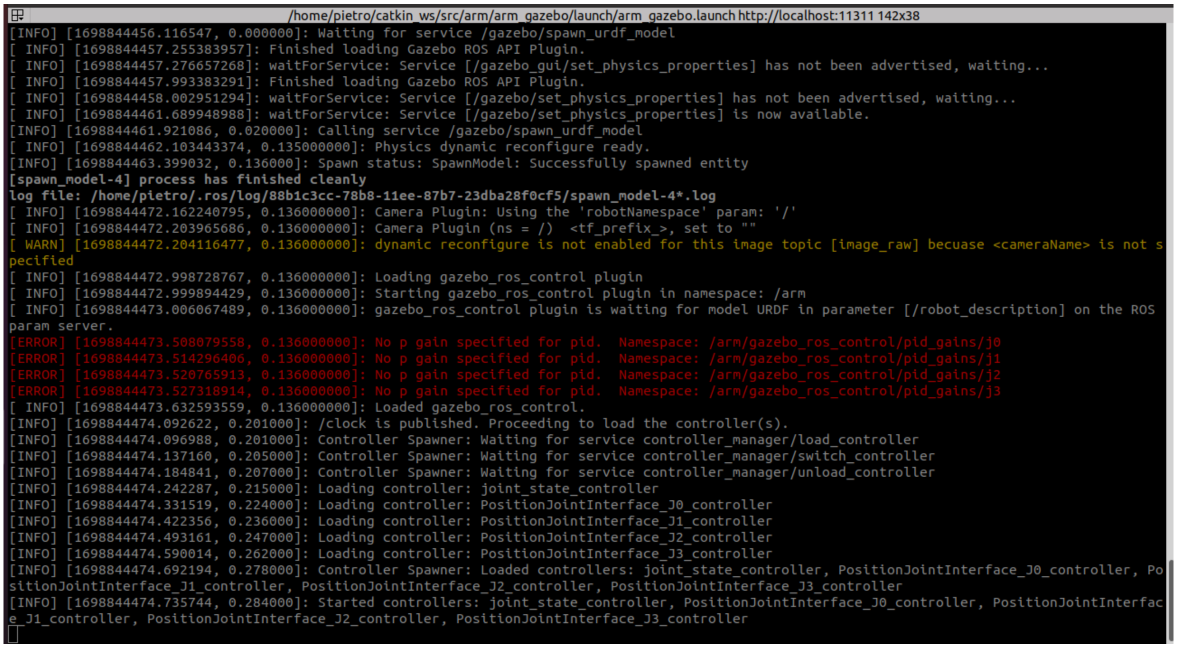
```
 * /arm/EffortJointInterface_trajectory_controller/stop_trajectory_duration: 0
 * /arm/EffortJointInterface_trajectory_controller/type: effort_controller...
 * /arm/PositionJointInterface_J0_controller/joint: j0
 * /arm/PositionJointInterface_J0_controller/type: position_controll...
 * /arm/PositionJointInterface_J1_controller/joint: j1
 * /arm/PositionJointInterface_J1_controller/type: position_controll...
 * /arm/PositionJointInterface_J2_controller/joint: j2
 * /arm/PositionJointInterface_J2_controller/type: position_controll...
 * /arm/PositionJointInterface_J3_controller/joint: j3
 * /arm/PositionJointInterface_J3_controller/type: position_controll...
 * /arm/PositionJointInterface_trajectory_controller/action_monitor_rate: 30
 * /arm/PositionJointInterface_trajectory_controller/constraints/goal_time: 0.5
 * /arm/PositionJointInterface_trajectory_controller/joints: ['j0', 'j1', 'j2'...
 * /arm/PositionJointInterface_trajectory_controller/state_publish_rate: 25
 * /arm/PositionJointInterface_trajectory_controller/stop_trajectory_duration: 0
 * /arm/PositionJointInterface_trajectory_controller/type: position_controll...
 * /arm/VelocityJointInterface_J0_controller/joint: j0
 * /arm/VelocityJointInterface_J0_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_J1_controller/joint: j1
 * /arm/VelocityJointInterface_J1_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_J2_controller/joint: j2
 * /arm/VelocityJointInterface_J2_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_J3_controller/joint: j3
 * /arm/VelocityJointInterface_J3_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_trajectory_controller/action_monitor_rate: 30
 * /arm/VelocityJointInterface_trajectory_controller/constraints/goal_time: 0.5
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/d: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/i: 15
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/p: 500
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/d: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/i: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/p: 200
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/d: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/i: 15
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/p: 65
```



```
 * /arm/VelocityJointInterface_J2_controller/joint: j2
 * /arm/VelocityJointInterface_J2_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_J3_controller/joint: j3
 * /arm/VelocityJointInterface_J3_controller/type: velocity_controll...
 * /arm/VelocityJointInterface_trajectory_controller/action_monitor_rate: 30
 * /arm/VelocityJointInterface_trajectory_controller/constraints/goal_time: 0.5
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/d: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/i: 15
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j0/p: 500
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/d: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/i: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j1/p: 200
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/d: 10
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/i: 15
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j2/p: 65
 * /arm/VelocityJointInterface_trajectory_controller/gains/j3/d: 7
 * /arm/VelocityJointInterface_trajectory_controller/gains/j3/i: 12
 * /arm/VelocityJointInterface_trajectory_controller/gains/j3/i_clamp: 30
 * /arm/VelocityJointInterface_trajectory_controller/gains/j3/p: 31
 * /arm/VelocityJointInterface_trajectory_controller/joints: ['j0', 'j1', 'j2'...
 * /arm/VelocityJointInterface_trajectory_controller/state_publish_rate: 25
 * /arm/VelocityJointInterface_trajectory_controller/stop_trajectory_duration: 0
 * /arm/VelocityJointInterface_trajectory_controller/type: velocity_controll...
 * /arm/joint_state_controller/publish_rate: 100
 * /arm/joint_state_controller/type: joint_state_contr...
 * /arm/robot_state_publisher/publish_frequency: 100
 * /gazebo/enable_ros_network: True
 * /robot_description: <?xml version="1....
 * /rosdistro: noetic
 * /rosversion: 1.16.0
 * /use_sim_time: True

NODES
  /
    gazebo (gazebo_ros/gzserver)
```

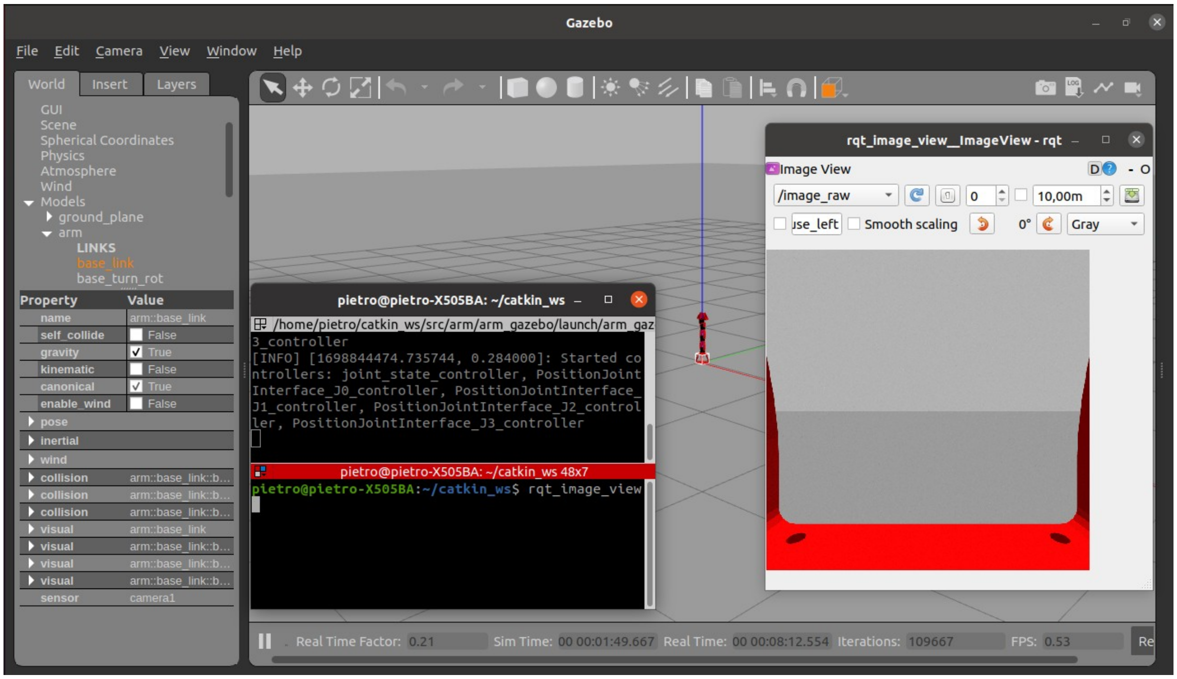


```
 * /rosversion: 1.16.0
 * /use_sim_time: True

NODES
  /
    gazebo (gazebo_ros/gzserver)
    gazebo_gui (gazebo_ros/gzclient)
    spawn_model (gazebo_ros/spawn_model)
  /arm/
    controller_spawner (controller_manager/spawner)
    robot_state_publisher (robot_state_publisher/robot_state_publisher)

auto-starting new master
process[master]: started with pid [31375]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 88b1c3cc-78b8-11ee-87b7-23dba28f0cf5
process[rosout-1]: started with pid [31385]
started core service [/rosout]
process[gazebo-2]: started with pid [31392]
process[gazebo_gui-3]: started with pid [31394]
process[spawn_model-4]: started with pid [31396]
process[arm/controller_spawner-5]: started with pid [31400]
process[arm/robot_state_publisher-6]: started with pid [31404]
[ WARN] [1698844450.615141594]: The root link base_link has an inertia specified in the URDF, but KDL does not support a root link with an ine
rtia.  As a workaround, you can add an extra dummy link to your URDF.
[INFO] [1698844455.946215, 0.000000]: Waiting for /clock to be available...
[INFO] [1698844456.083539, 0.000000]: Loading model XML from ros parameter robot_description
[INFO] [1698844456.116547, 0.000000]: Waiting for service /gazebo/spawn_urdf_model
[ INFO] [1698844457.255383957]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1698844457.276657268]: waitForService: Service [/gazebo_gui/set_physics_properties] has not been advertised, waiting...
[ INFO] [1698844457.993383291]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1698844458.002951294]: waitForService: Service [/gazebo/set_physics_properties] has not been advertised, waiting...
[ INFO] [1698844461.689948988]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1698844461.921086, 0.020000]: Calling service /gazebo/spawn_urdf_model
[ INFO] [1698844462.103443374, 0.135000000]: Physics dynamic reconfigure ready.
[INFO] [1698844463.399032, 0.136000]: Spawn status: SpawnModel: Successfully spawned entity
[spawn_model-4] process has finished cleanly
```



/home/pietro/catkin_ws/src/arm/arm_gazebo/launch/arm_gazebo.launch http://localhost:11311 142x38

```
[INFO] [1698844456.116547, 0.000000]: Waiting for service /gazebo/spawn_urdf_model
[ INFO] [1698844457.255383957]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1698844457.276657268]: waitForService: Service [/gazebo_gui/set_physics_properties] has not been advertised, waiting...
[ INFO] [1698844457.993383291]: Finished loading Gazebo ROS API Plugin.
[ INFO] [1698844458.002951294]: waitForService: Service [/gazebo/set_physics_properties] has not been advertised, waiting...
[ INFO] [1698844461.689948988]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1698844461.921086, 0.020000]: Calling service /gazebo/spawn_urdf_model
[ INFO] [1698844462.103443374, 0.135000000]: Physics dynamic reconfigure ready.
[INFO] [1698844463.399032, 0.136000]: Spawn status: SpawnModel: Successfully spawned entity
[spawn_model-4] process has finished cleanly
log file: /home/pietro/.ros/log/88b1c3cc-78b8-11ee-87b7-23dba28f0cf5/spawn_model-4*.log
[ INFO] [1698844472.162240795, 0.136000000]: Camera Plugin: Using the 'robotNamespace' param: '/'
[ INFO] [1698844472.203965686, 0.136000000]: Camera Plugin (ns = /)  <tf_prefix_>, set to ""
[ WARN] [1698844472.204116477, 0.136000000]: dynamic reconfigure is not enabled for this image topic [image_raw] becuase <cameraName> is not s
pecified
[ INFO] [1698844472.998728767, 0.136000000]: Loading gazebo_ros_control plugin
[ INFO] [1698844472.999894429, 0.136000000]: Starting gazebo_ros_control plugin in namespace: /arm
[ INFO] [1698844473.006067489, 0.136000000]: gazebo_ros_control plugin is waiting for model URDF in parameter [/robot_description] on the ROS
param server.
[ERROR] [1698844473.508079558, 0.136000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j0
[ERROR] [1698844473.514296406, 0.136000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j1
[ERROR] [1698844473.520765913, 0.136000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j2
[ERROR] [1698844473.527318914, 0.136000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j3
[ INFO] [1698844473.632593559, 0.136000000]: Loaded gazebo_ros_control.
[INFO] [1698844474.092622, 0.201000]: /clock is published. Proceeding to load the controller(s).
[INFO] [1698844474.096988, 0.201000]: Controller Spawner: Waiting for service controller_manager/load_controller
[INFO] [1698844474.137160, 0.205000]: Controller Spawner: Waiting for service controller_manager/switch_controller
[INFO] [1698844474.184841, 0.207000]: Controller Spawner: Waiting for service controller_manager/unload_controller
[INFO] [1698844474.242287, 0.215000]: Loading controller: joint_state_controller
[INFO] [1698844474.331519, 0.224000]: Loading controller: PositionJointInterface_J0_controller
[INFO] [1698844474.422356, 0.236000]: Loading controller: PositionJointInterface_J1_controller
[INFO] [1698844474.493161, 0.247000]: Loading controller: PositionJointInterface_J2_controller
[INFO] [1698844474.590014, 0.262000]: Loading controller: PositionJointInterface_J3_controller
[INFO] [1698844474.692194, 0.278000]: Controller Spawner: Loaded controllers: joint_state_controller, PositionJointInterface_J0_controller, Po
sitionJointInterface_J1_controller, PositionJointInterface_J2_controller, PositionJointInterface_J3_controller
[INFO] [1698844474.735744, 0.284000]: Started controllers: joint_state_controller, PositionJointInterface_J0_controller, PositionJointInterfac
e_J1_controller, PositionJointInterface_J2_controller, PositionJointInterface_J3_controller
```



| Property | Value |
|---|---|
| name | arm::base_link |
| self_collide | □ False |
| gravity | ☑ True |
| kinematic | □ False |
| canonical | ☑ True |
| enable_wind | □ False |
| pose | |
| inertial | |
| wind | |
| collision | arm::base_link::b... |
| collision | arm::base_link::b... |
| collision | arm::base_link::b... |
| visual | arm::base_link |
| visual | arm::base_link::b... |
| visual | arm::base_link::b... |
| visual | arm::base_link::b... |
| sensor | camera1 |

pietro@pietro-X505BA: ~/catkin_ws

/home/pietro/catkin_ws/src/arm/arm_gazebo/launch/arm_gaz

```
3_controller
[INFO] [1698844474.735744, 0.284000]: Started co
ntrollers: joint_state_controller, PositionJoint
Interface_J0_controller, PositionJointInterface_
J1_controller, PositionJointInterface_J2_control
ler, PositionJointInterface_J3_controller
```

pietro@pietro-X505BA: ~/catkin_ws 48x7

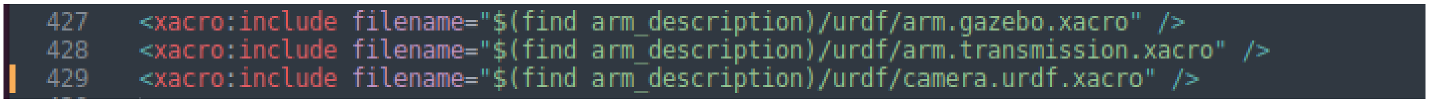
```
pietro@pietro-X505BA:~/catkin_ws$ rqt_image_view
```

/image_raw | 0 | □ 10,00m | □ ıse_left | □ Smooth scaling | 0° | Gray

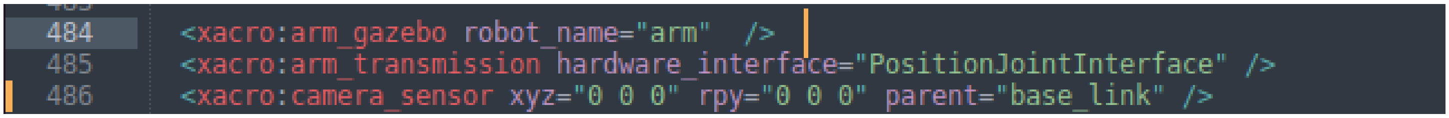
Real Time Factor: 0.21 Sim Time: 00 00:01:49.667 Real Time: 00 00:08:12.554 Iterations: 109667 FPS: 0.53 Re

3.d **Optionally:** *You can create a camera.xacro file (or download one from https://github.com/CentroEPiaggio/irobotcreate2ros/blob/master/model/camera.urdf.xacro) and add it to your robot URDF using <xacro:include>*

I downloaded the code camera.urdf.xacro from the link and after I put it in the arm_description/urdf. Then if i want use this camera.urdf.xacro, i must to commenting the lines regarding link name = "camera_link" from 403 line to 425 line in arm.urdf.xacro and decommenting the xacro:include regarding camera.urdf.xacro below and xacro:camera_sensor and:

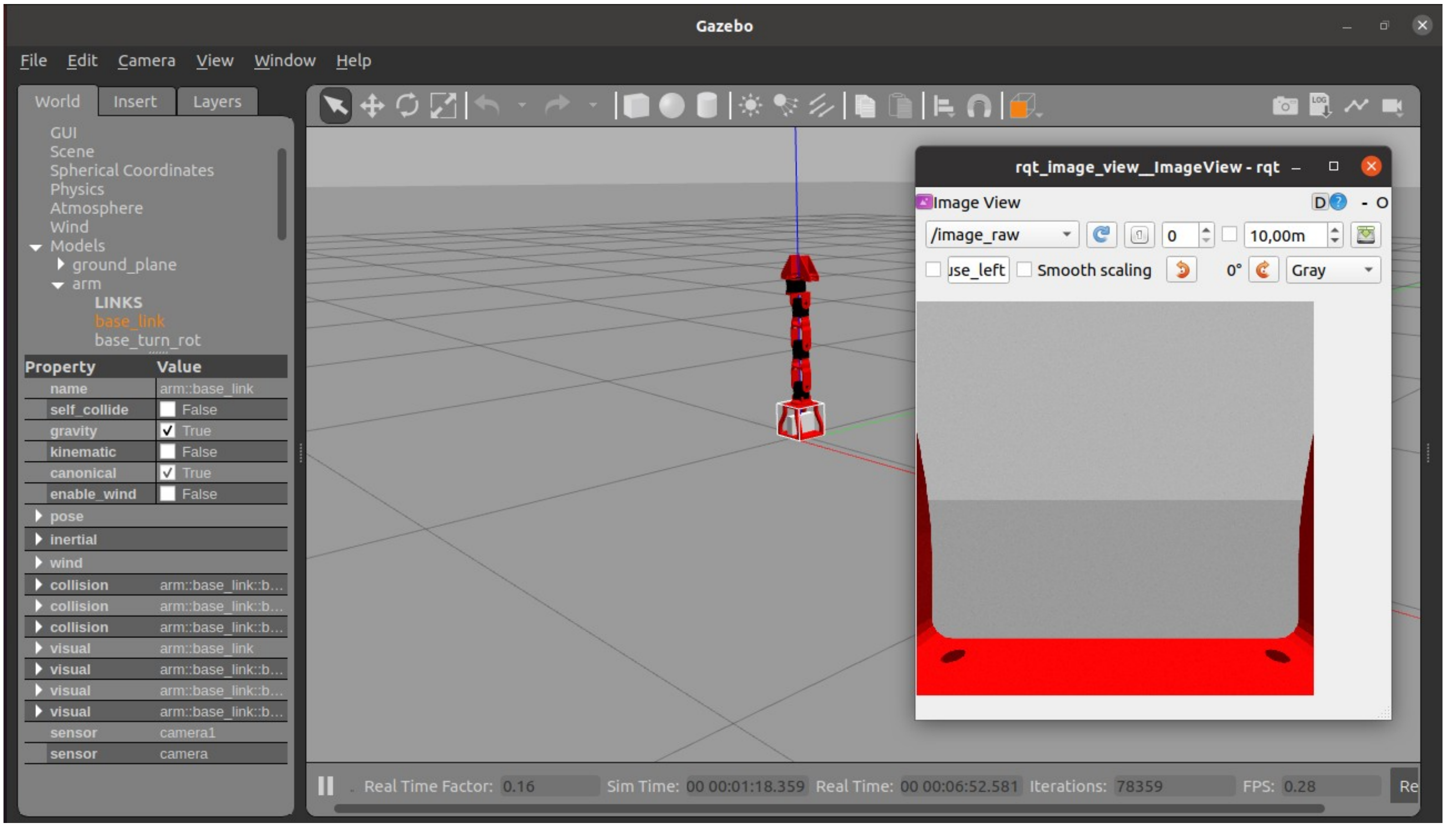


```
<xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />
<xacro:include filename="$(find arm_description)/urdf/arm.transmission.xacro" />
<xacro:include filename="$(find arm_description)/urdf/camera.urdf.xacro" />
```



```
<xacro:arm_gazebo robot_name="arm"  />
<xacro:arm_transmission hardware_interface="PositionJointInterface" />
<xacro:camera_sensor xyz="0 0 0" rpy="0 0 0" parent="base_link" />
```

*"roslaunch arm_gazebo arm_gazebo.launch"***by terminal** *and "rqt_image_view"* **by another terminal***:*

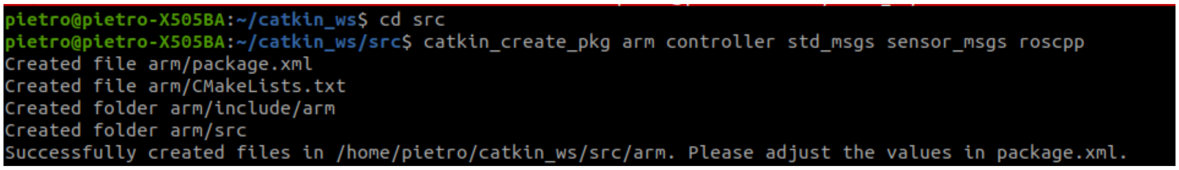


| Property | Value |
|---|---|
| name | arm::base_link |
| self_collide | □ False |
| gravity | ☑ True |
| kinematic | □ False |
| canonical | ☑ True |
| enable_wind | □ False |
| pose | |
| inertial | |
| wind | |
| collision | arm::base_link::b... |
| collision | arm::base_link::b... |
| collision | arm::base_link::b... |
| visual | arm::base_link |
| visual | arm::base_link::b... |
| visual | arm::base_link::b... |
| visual | arm::base_link::b... |
| sensor | camera1 |
| sensor | camera |

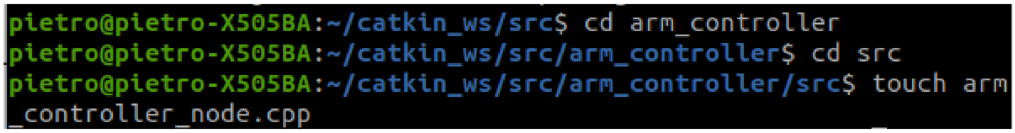
Real Time Factor: 0.16 Sim Time: 00 00:01:18.359 Real Time: 00 00:06:52.581 Iterations: 78359 FPS: 0.28

4 Create a ROS publisher node that reads the joint state and sends joint position commands to your robot

4.a Create an *arm_controller* package with a ROS C++ node named *arm_controller_node*. The dependencies are *roscpp*, *sensor_msgs* and *std_msgs*. Modify opportunely the *CMakeLists.txt* file to compile your node. **Hint:** uncomment *add_executable* and *target_link_libraries* lines

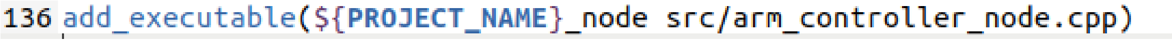


```
pietro@pietro-X505BA:~/catkin_ws$ cd src
pietro@pietro-X505BA:~/catkin_ws/src$ catkin_create_pkg arm controller std_msgs sensor_msgs roscpp
Created file arm/package.xml
Created file arm/CMakeLists.txt
Created folder arm/include/arm
Created folder arm/src
Successfully created files in /home/pietro/catkin_ws/src/arm. Please adjust the values in package.xml.
```

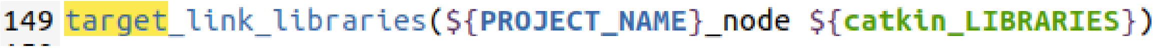


```
pietro@pietro-X505BA:~/catkin_ws/src$ cd arm_controller
pietro@pietro-X505BA:~/catkin_ws/src/arm_controller$ cd src
pietro@pietro-X505BA:~/catkin_ws/src/arm_controller/src$ touch arm
_controller_node.cpp
```

**CmakeLists:**



```
add_executable(${PROJECT_NAME}_node src/arm_controller_node.cpp)
```

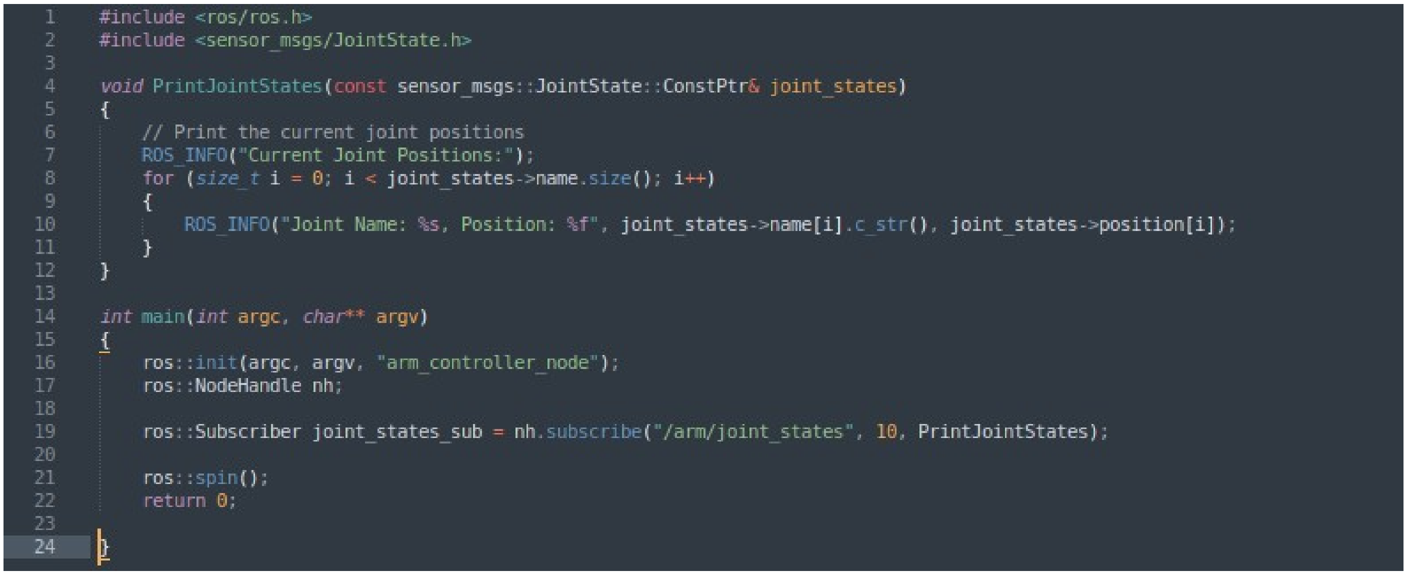


```
target_link_libraries(${PROJECT_NAME}_node ${catkin_LIBRARIES})
```

4.b Create a subscriber to the topic *joint_states* and a callback function that prints the current joint positions (see Slide 45). **Note:** the topic contains a *sensor_msgs/JointState*

Create publishers that write commands onto the controllers' */command* topics (see Slide 46). **Note:** the command is a *std_msgs/Float64*

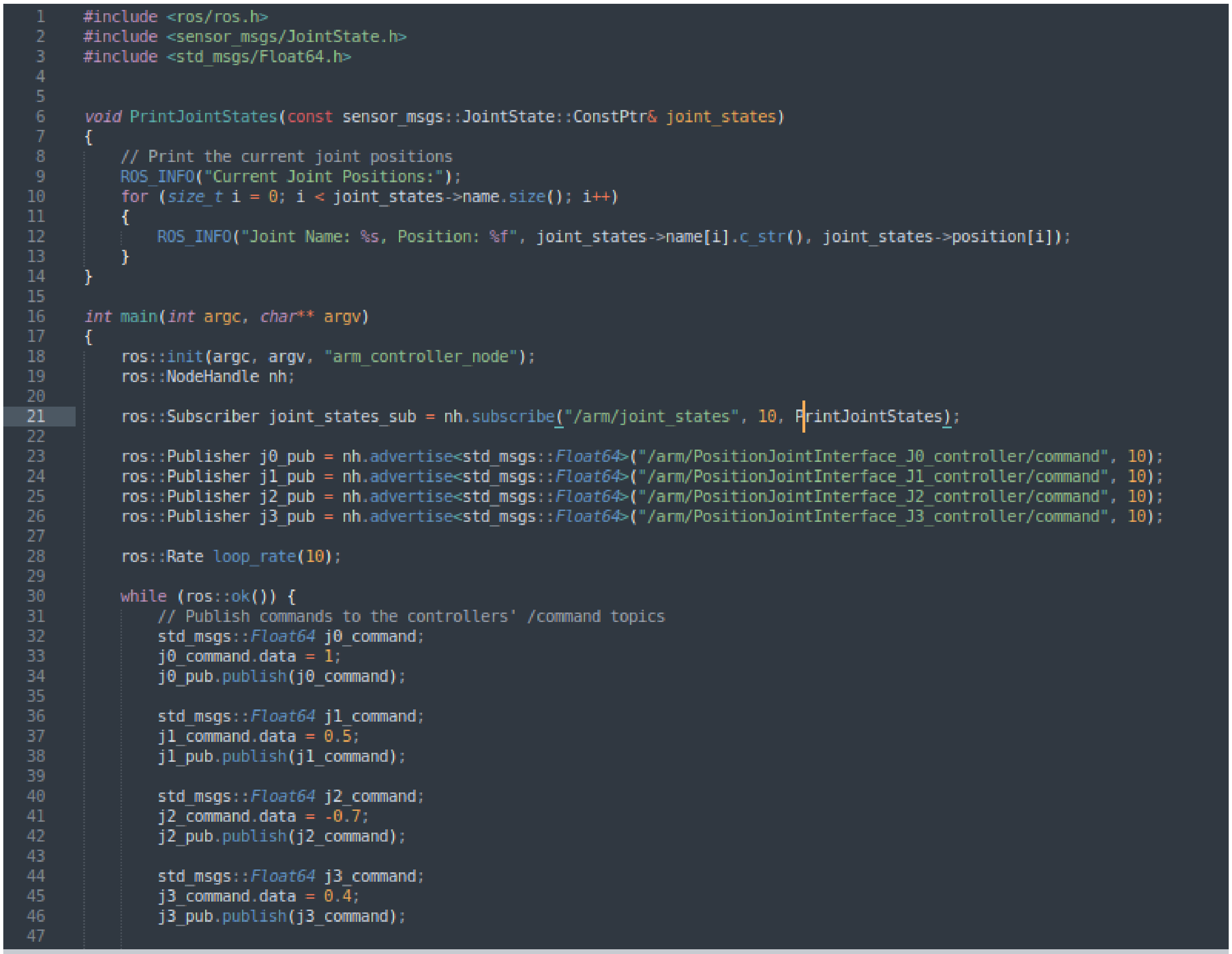
**Subscriber:**



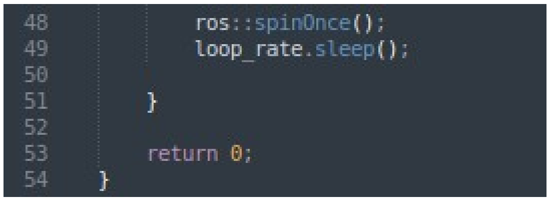
```
#include <ros/ros.h>
#include <sensor_msgs/JointState.h>

void PrintJointStates(const sensor_msgs::JointState::ConstPtr& joint_states)
{
    // Print the current joint positions
    ROS_INFO("Current Joint Positions:");
    for (size_t i = 0; i < joint_states->name.size(); i++)
    {
        ROS_INFO("Joint Name: %s, Position: %f", joint_states->name[i].c_str(), joint_states->position[i]);
    }
}

int main(int argc, char** argv)
{
    ros::init(argc, argv, "arm_controller_node");
    ros::NodeHandle nh;

    ros::Subscriber joint_states_sub = nh.subscribe("/arm/joint_states", 10, PrintJointStates);

    ros::spin();
    return 0;

}
```

**Subscriber and Publisher:** *into file arm_controller_node.cpp:*

*this node controls a robot by publishing position commands to the robot's joints and monitors the actual joint positions through messages on the /arm/joint_states topic. The control frequency is set to 10 Hz. A subscriber is created to listen for messages on the /arm/joint_states topic, and when a message is received, it calls the PrintJointStates function. Four publishers are created to send commands to the robot's four joints through their respective command topics, such as*

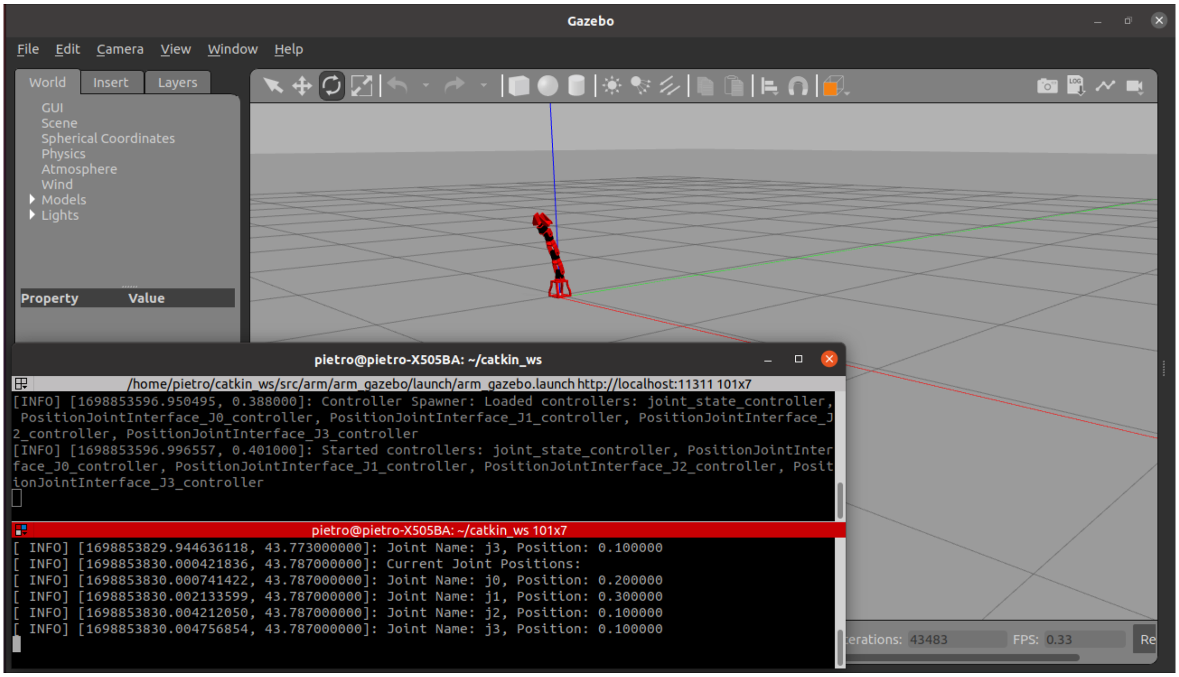


```cpp
#include <ros/ros.h>
#include <sensor_msgs/JointState.h>
#include <std_msgs/Float64.h>


void PrintJointStates(const sensor_msgs::JointState::ConstPtr& joint_states)
{
    // Print the current joint positions
    ROS_INFO("Current Joint Positions:");
    for (size_t i = 0; i < joint_states->name.size(); i++)
    {
        ROS_INFO("Joint Name: %s, Position: %f", joint_states->name[i].c_str(), joint_states->position[i]);
    }
}

int main(int argc, char** argv)
{
    ros::init(argc, argv, "arm_controller_node");
    ros::NodeHandle nh;

    ros::Subscriber joint_states_sub = nh.subscribe("/arm/joint_states", 10, PrintJointStates);

    ros::Publisher j0_pub = nh.advertise<std_msgs::Float64>("/arm/PositionJointInterface_J0_controller/command", 10);
    ros::Publisher j1_pub = nh.advertise<std_msgs::Float64>("/arm/PositionJointInterface_J1_controller/command", 10);
    ros::Publisher j2_pub = nh.advertise<std_msgs::Float64>("/arm/PositionJointInterface_J2_controller/command", 10);
    ros::Publisher j3_pub = nh.advertise<std_msgs::Float64>("/arm/PositionJointInterface_J3_controller/command", 10);

    ros::Rate loop_rate(10);

    while (ros::ok()) {
        // Publish commands to the controllers' /command topics
        std_msgs::Float64 j0_command;
        j0_command.data = 1;
        j0_pub.publish(j0_command);

        std_msgs::Float64 j1_command;
        j1_command.data = 0.5;
        j1_pub.publish(j1_command);

        std_msgs::Float64 j2_command;
        j2_command.data = -0.7;
        j2_pub.publish(j2_command);

        std_msgs::Float64 j3_command;
        j3_command.data = 0.4;
        j3_pub.publish(j3_command);

```



```cpp
        ros::spinOnce();
        loop_rate.sleep();

    }

    return 0;
}
```

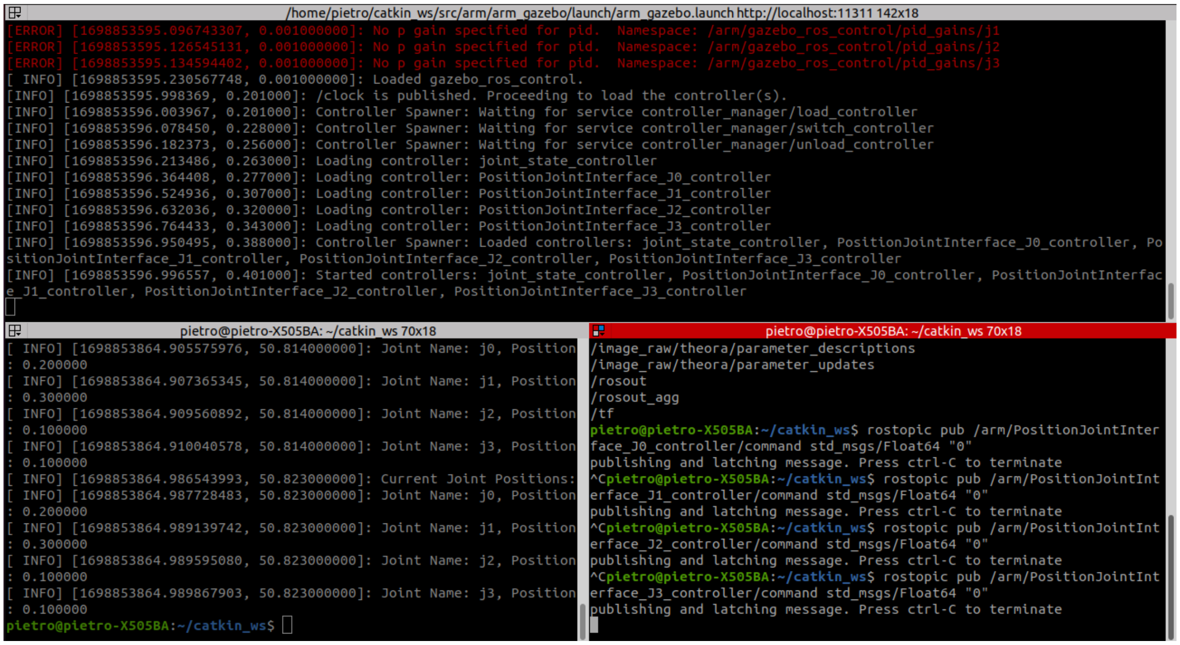
**check Gazebo:**



```
pietro@pietro-X505BA: ~/catkin_ws
/home/pietro/catkin_ws/src/arm/arm_gazebo/launch/arm_gazebo.launch http://localhost:11311 101x7
[INFO] [1698853596.950495, 0.388000]: Controller Spawner: Loaded controllers: joint_state_controller,
 PositionJointInterface_J0_controller, PositionJointInterface_J1_controller, PositionJointInterface_J
2_controller, PositionJointInterface_J3_controller
[INFO] [1698853596.996557, 0.401000]: Started controllers: joint_state_controller, PositionJointInter
face_J0_controller, PositionJointInterface_J1_controller, PositionJointInterface_J2_controller, Posit
ionJointInterface_J3_controller
```

```
pietro@pietro-X505BA: ~/catkin_ws 101x7
[ INFO] [1698853829.944636118, 43.773000000]: Joint Name: j3, Position: 0.100000
[ INFO] [1698853830.000421836, 43.787000000]: Current Joint Positions:
[ INFO] [1698853830.000741422, 43.787000000]: Joint Name: j0, Position: 0.200000
[ INFO] [1698853830.002133599, 43.787000000]: Joint Name: j1, Position: 0.300000
[ INFO] [1698853830.004212050, 43.787000000]: Joint Name: j2, Position: 0.100000
[ INFO] [1698853830.004756854, 43.787000000]: Joint Name: j3, Position: 0.100000
```

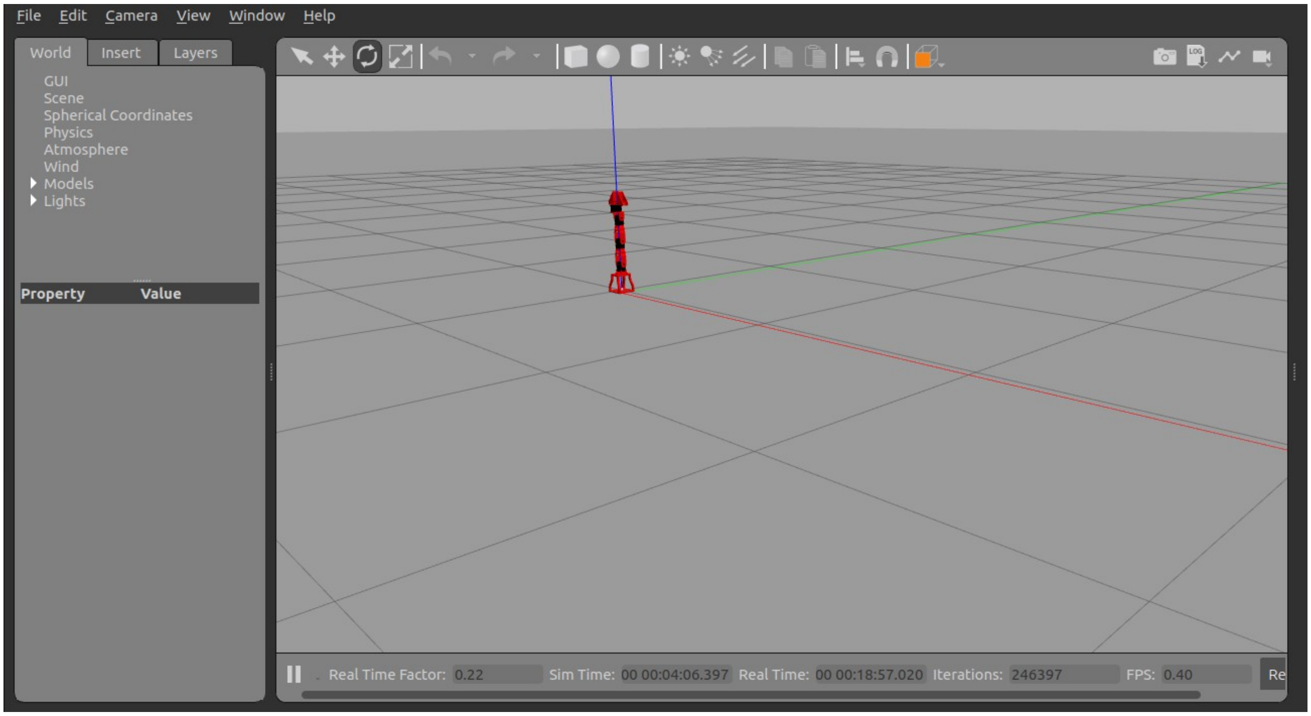
I used by terminal the command “rostopic pub /topic std_msgs/Float64 “0” “ to return the robot to the starting posizion



```
/home/pietro/catkin_ws/src/arm/arm_gazebo/launch/arm_gazebo.launch http://localhost:11311 142x18
[ERROR] [1698853595.096743307, 0.001000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j1
[ERROR] [1698853595.126545131, 0.001000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j2
[ERROR] [1698853595.134594402, 0.001000000]: No p gain specified for pid.  Namespace: /arm/gazebo_ros_control/pid_gains/j3
[ INFO] [1698853595.230567748, 0.001000000]: Loaded gazebo_ros_control.
[INFO] [1698853595.998369, 0.201000]: /clock is published. Proceeding to load the controller(s).
[INFO] [1698853596.003967, 0.201000]: Controller Spawner: Waiting for service controller_manager/load_controller
[INFO] [1698853596.078450, 0.228000]: Controller Spawner: Waiting for service controller_manager/switch_controller
[INFO] [1698853596.182373, 0.256000]: Controller Spawner: Waiting for service controller_manager/unload_controller
[INFO] [1698853596.213486, 0.263000]: Loading controller: joint_state_controller
[INFO] [1698853596.364408, 0.277000]: Loading controller: PositionJointInterface_J0_controller
[INFO] [1698853596.524936, 0.307000]: Loading controller: PositionJointInterface_J1_controller
[INFO] [1698853596.632036, 0.320000]: Loading controller: PositionJointInterface_J2_controller
[INFO] [1698853596.764433, 0.343000]: Loading controller: PositionJointInterface_J3_controller
[INFO] [1698853596.950495, 0.388000]: Controller Spawner: Loaded controllers: joint_state_controller, PositionJointInterface_J0_controller, Po
sitionJointInterface_J1_controller, PositionJointInterface_J2_controller, PositionJointInterface_J3_controller
[INFO] [1698853596.996557, 0.401000]: Started controllers: joint_state_controller, PositionJointInterface_J0_controller, PositionJointInterfac
e_J1_controller, PositionJointInterface_J2_controller, PositionJointInterface_J3_controller
```

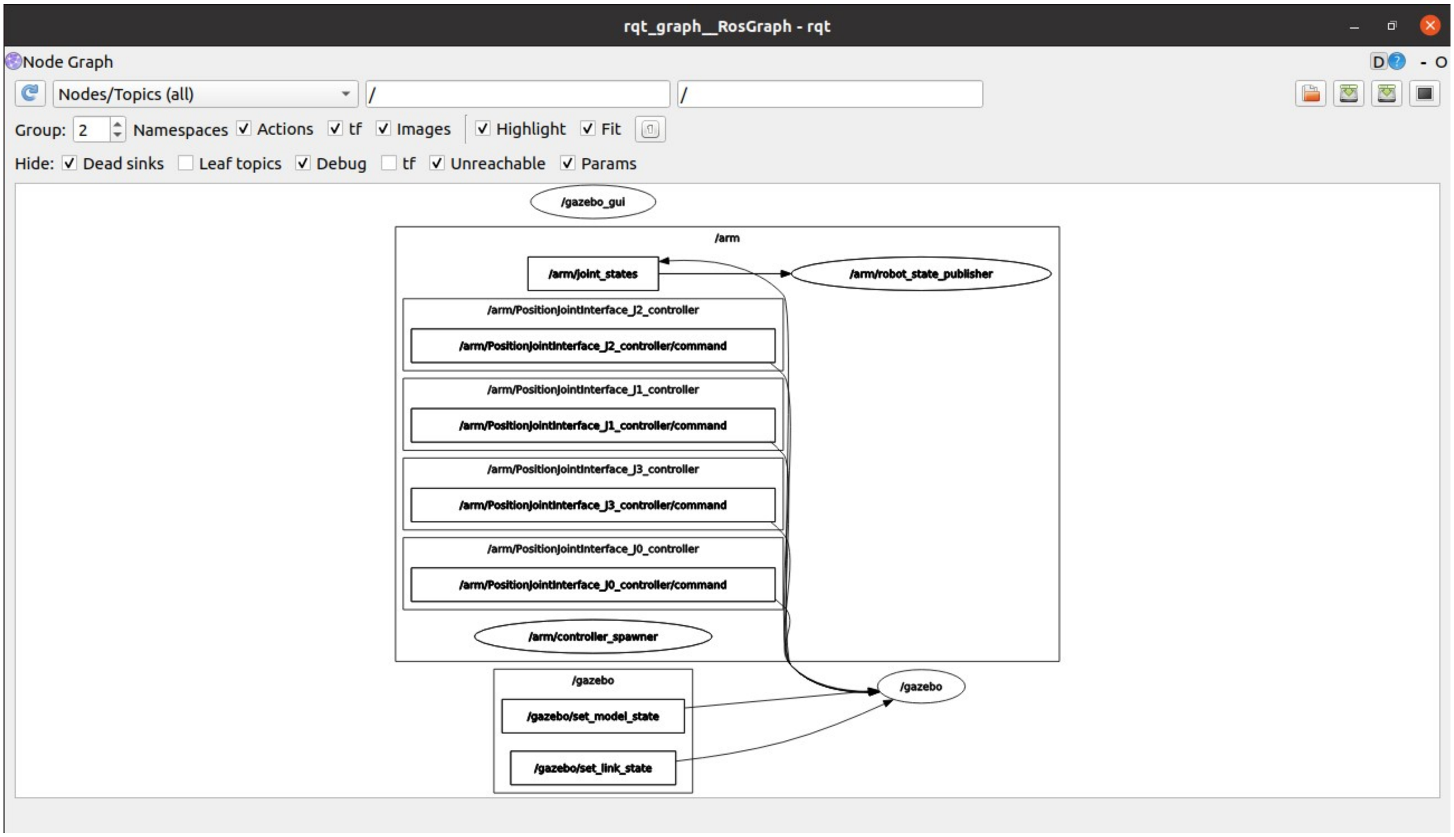
```
pietro@pietro-X505BA: ~/catkin_ws 70x18
[ INFO] [1698853864.905575976, 50.814000000]: Joint Name: j0, Position
: 0.200000
[ INFO] [1698853864.907365345, 50.814000000]: Joint Name: j1, Position
: 0.300000
[ INFO] [1698853864.909560892, 50.814000000]: Joint Name: j2, Position
: 0.100000
[ INFO] [1698853864.910040578, 50.814000000]: Joint Name: j3, Position
: 0.100000
[ INFO] [1698853864.986543993, 50.823000000]: Current Joint Positions:
[ INFO] [1698853864.987728483, 50.823000000]: Joint Name: j0, Position
: 0.200000
[ INFO] [1698853864.989139742, 50.823000000]: Joint Name: j1, Position
: 0.300000
[ INFO] [1698853864.989595080, 50.823000000]: Joint Name: j2, Position
: 0.100000
[ INFO] [1698853864.989867903, 50.823000000]: Joint Name: j3, Position
: 0.100000
pietro@pietro-X505BA:~/catkin_ws$
```

```
pietro@pietro-X505BA: ~/catkin_ws 70x18
/image_raw/theora/parameter_descriptions
/image_raw/theora/parameter_updates
/rosout
/rosout_agg
/tf
pietro@pietro-X505BA:~/catkin_ws$ rostopic pub /arm/PositionJointInter
face_J0_controller/command std_msgs/Float64 "0"
publishing and latching message. Press ctrl-C to terminate
^Cpietro@pietro-X505BA:~/catkin_ws$ rostopic pub /arm/PositionJointInt
erface_J1_controller/command std_msgs/Float64 "0"
publishing and latching message. Press ctrl-C to terminate
^Cpietro@pietro-X505BA:~/catkin_ws$ rostopic pub /arm/PositionJointInt
erface_J2_controller/command std_msgs/Float64 "0"
publishing and latching message. Press ctrl-C to terminate
^Cpietro@pietro-X505BA:~/catkin_ws$ rostopic pub /arm/PositionJointInt
erface_J3_controller/command std_msgs/Float64 "0"
publishing and latching message. Press ctrl-C to terminate
```



Real Time Factor: 0.22 Sim Time: 00 00:04:06.397 Real Time: 00 00:18:57.020 Iterations: 246397 FPS: 0.40 Re

*command “rqt_graph”* **by terminal**


rqt_graph__RosGraph - rqt
Node Graph
/gazebo_gui
/arm
/arm/joint_states
/arm/robot_state_publisher
/arm/PositionJointInterface_J2_controller
/arm/PositionJointInterface_J2_controller/command
/arm/PositionJointInterface_J1_controller
/arm/PositionJointInterface_J1_controller/command
/arm/PositionJointInterface_J3_controller
/arm/PositionJointInterface_J3_controller/command
/arm/PositionJointInterface_J0_controller
/arm/PositionJointInterface_J0_controller/command
/arm/controller_spawner
/gazebo
/gazebo/set_model_state
/gazebo/set_link_state
/gazebo

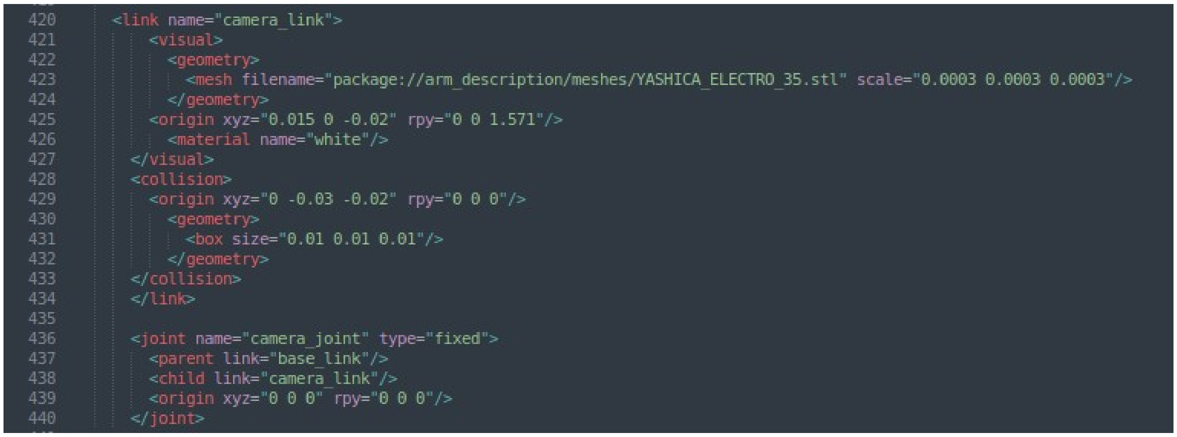
Nodes/Topics (all) / /

Group: 2 Namespaces ☑ Actions ☑ tf ☑ Images ☑ Highlight ☑ Fit

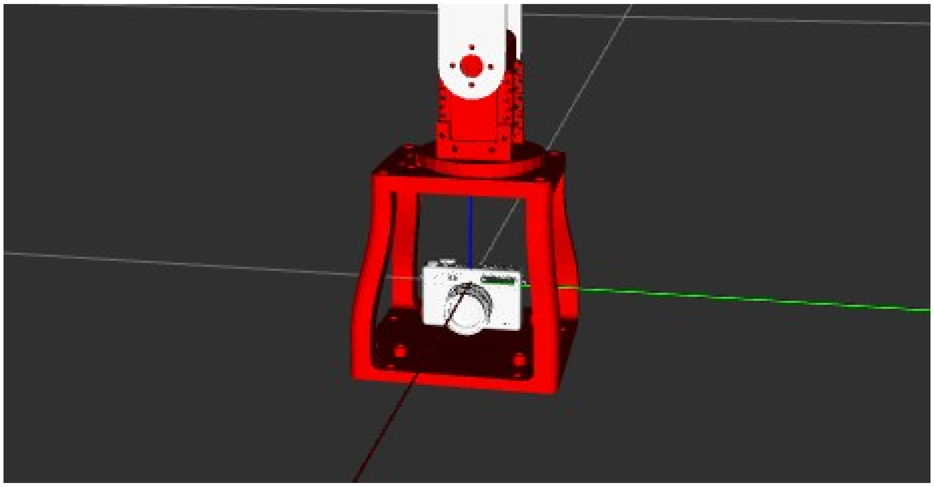
Hide: ☑ Dead sinks □ Leaf topics ☑ Debug □ tf ☑ Unreachable ☑ Params

# EXTRA

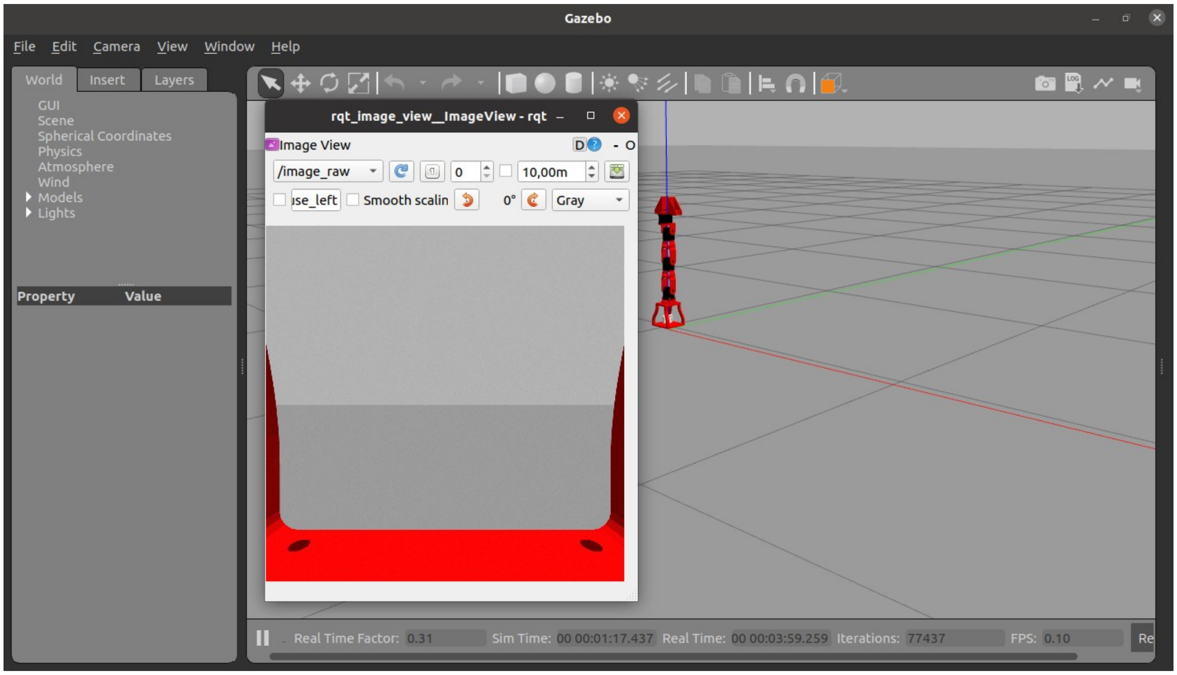
*I downloaded the file ".stl" regarding a real camera named "YASHICA_ELECTRO_35" from Internet and I put it in the mesh folder of arm_description and I substitued it in place of the camera_link added at point (a) of number 3 in arm.urdf.xacro*



```
<link name="camera_link">
    <visual>
      <geometry>
        <mesh filename="package://arm_description/meshes/YASHICA_ELECTRO_35.stl" scale="0.0003 0.0003 0.0003"/>
      </geometry>
    <origin xyz="0.015 0 -0.02" rpy="0 0 1.571"/>
      <material name="white"/>
  </visual>
  <collision>
    <origin xyz="0 -0.03 -0.02" rpy="0 0 0"/>
      <geometry>
        <box size="0.01 0.01 0.01"/>
      </geometry>
  </collision>
  </link>

  <joint name="camera_joint" type="fixed">
    <parent link="base_link"/>
    <child link="camera_link"/>
    <origin xyz="0 0 0" rpy="0 0 0"/>
  </joint>
```



**Check Gazebo**: *I used the command "roslaunch arm_gazebo arm_gazebo.launch" by terminal and rqt_image_view by another terminal to verify that camera Yashica worked.*



/image_raw 0 ☐ 10,00m

☐ ıse_left ☐ Smooth scalin 0° Gray

Real Time Factor: 0.31 Sim Time: 00 00:01:17.437 Real Time: 00 00:03:59.259 Iterations: 77437 FPS: 0.10 Re